

LNCS 3093

Sokratis K. Katsikas
Stefanos Gritzalis
Javier Lopez (Eds.)

Public Key Infrastructure

**First European PKI Workshop:
Research and Applications, EuroPKI 2004
Samos Island, Greece, June 2004, Proceedings**



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

New York University, NY, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Sokratis K. Katsikas Stefanos Gritzalis
Javier Lopez (Eds.)

Public Key Infrastructure

First European PKI Workshop:
Research and Applications, EuroPKI 2004
Samos Island, Greece, June 25-26, 2004
Proceedings

Volume Editors

Sokratis K. Katsikas
University of the Aegean
Rector's Office, Administration Building
University Hill, GR-81100 Mytilene, Greece
E-mail: ska@aegean.gr

Stefanos Gritzalis
University of the Aegean
Department of Information and Communication Systems Engineering
Laboratory of Information and Communication Systems Security
Karlovassi, GR-83200 Samos, Greece
E-mail: sgritz@aegean.gr

Javier Lopez
University of Malaga
Computer Science Department, E.T.S. Ingeniería Informática
Campus de Teatinos, Spain
E-mail: jlm@lcc.uma.es

Library of Congress Control Number: 2004107465

CR Subject Classification (1998): E.3, D.4.6, C.2.0, F.2.1, H.3, H.4, K.4.4, K.6.5

ISSN 0302-9743

ISBN 3-540-22216-2 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable to prosecution under the German Copyright Law.

Springer-Verlag is a part of Springer Science+Business Media
springeronline.com

© Springer-Verlag Berlin Heidelberg 2004
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 11012214 06/3142 5 4 3 2 1 0

Preface

There is no doubt that the Internet is affecting every aspect of our lives; the most significant changes are occurring in private and public sector organizations that are transforming their conventional operating models to Internet-based service models, known as eBusiness, eCommerce and eGovernment. Companies, institutions and organizations, irrespective of their size, are nowadays utilizing the Internet for communicating with their customers, suppliers and partners; for facilitating the interconnection of their employees and branches; for connecting to their back-end data systems and for performing commercial transactions. In such an environment, where almost every organization relies heavily on information and communications technologies, new dependencies and risks arise. *Public Key Infrastructure (PKI)* is probably one of the most important items in the arsenal of security measures that can be brought to bear against the aforementioned growing risks and threats.

PKI research has been active for more than 26 years. In 1978 R.L. Rivest, A. Shamir and L. Adleman published what is now commonly called the RSA cryptosystem (*Communications of the ACM*, Vol.21, No.2, pp.120–128, 1978), one of the most significant discoveries in the history of cryptography. Since the mathematical foundation of RSA rests on the intractability of factoring large composite integers, in the same year, R. Merkle demonstrated that certain computational puzzles could also be used in constructing public key cryptography (*Communications of the ACM*, Vol.21, No.4, pp.194–299, 1978).

As the years passed by, several countries started developing their PKI. Inevitably, several practical problems were identified. Although adhering to international standards, such as ITU, ISO, IETF and PKCS, different PKI systems (national and/or international) could not connect to one another. Subsequently, a number of organizations were formed to promote and support the interoperability of different PKIs between certain countries. Indicative examples of such organizations today include the *PKI Forum*, the *EESSI – European Electronic Signature Standardization Initiative* and the *Asia PKI Forum*.

To foster and stimulate these discussions in a research environment, the *International Workshops for Asian PKI (IWAP)* and the *US PKI Research Workshops* have been held annually since 2001 (IWAP 2001 in Korea, IWAP 2002 in Taiwan, IWAP 2004 in Japan) and since 2002 (the annual US PKI Research Workshops, hosted by the NIST) respectively. Their goal is to provide a framework for both theoreticians and practitioners to share their experience and research outcomes concerning good practices in applying PKI and related supporting technologies, together with prudent assessment and comparison of the technologies.

The first *European PKI Workshop: Research and Applications (EuroPKI 2004)* initiated a series of corresponding workshop activities in Europe. The EuroPKI 2004 workshop was held on 25–26 June 2004, on Samos Island, Greece, and was hosted by the University of the Aegean, Department of Information and Communication Systems Engineering, Laboratory of Information and Communication Systems Security (*Info-Sec-Lab*, www.icsd.aegean.gr/Info-Sec-Lab).

In response to the EuroPKI 2004 call for papers, 73 papers were submitted, whose authors came from 25 countries. Each paper was reviewed by three members of the Program Committee, on the basis of the significance, novelty, technical quality and PKI relevance of the work reported therein. At the end of the reviewing process, only 25 papers were selected for presentation, whose authors came from 13 countries, resulting in an acceptance rate of 34%. This volume contains these papers as well as 5 additional short papers.

We would like to thank all the members of the Program Committee, as well as the external reviewers, for their constructive and insightful comments during the review process. Moreover, we would like to express our gratitude to the members of the Organizing Committee for their continuous and valuable support. We also wish to express our thanks to Alfred Hofmann and his colleagues from Springer-Verlag, for their co-operation and their excellent work during the publication process. Finally, we would like to thank all the people who submitted their papers to the workshop, including those whose submissions were not selected for publication, and all the delegates from around the world who attended the first *European PKI Workshop*. Without their support the workshop would not have been possible.

June 2004

Sokratis K. Katsikas
Stefanos Gritzalis
Javier Lopez

EuroPKI'2004 Workshop Committee

General Chairman

Sokratis K. Katsikas University of the Aegean, Greece

Program Committee Co-Chairmen

Stefanos Gritzalis University of the Aegean, Greece

Javier Lopez University of Malaga, Spain

International Program Committee

Carlisle Adams	University of Ottawa, Canada
Giampaolo Bella	University of Catania, Italy
Ahto Buldas	Tallinn Technical University, Estonia
Mike Burmester	Florida State University, USA
Luke O'Connor	IBM, Switzerland
Sabrina De Capitani di Vimercati	University of Milan, Italy
Vassilios Chryssikopoulos	Ionian University, Greece
Ed Dawson	Queensland University of Technology, Australia
Yves Deswarte	LAAS-CNRS, France
Stephen Farrell	Trinity College Dublin, Ireland
Simon Foley	University College Corke, Ireland
Jordi Forne	Polytechnic University of Catalonia, Spain
Steven Furnell	University of Plymouth, UK
Dieter Gollmann	TU Hamburg-Harburg, Germany
Antonio Gomez-Skarmeta	University of Murcia, Spain
Dimitris Gritzalis	Athens University of Economics and Business, Greece
Hideki Imai	University of Tokio, Japan
Sushil Jajodia	George Mason University, USA
Kwangjo Kim	Information and Communications University, Korea
Spyros Kokolakis	University of the Aegean, Greece
Constantinos Lambrinoudakis	University of the Aegean, Greece
Dimitris Lekkas	University of the Aegean, Greece
Peter Lipp	Technical University of Graz, Austria
Jose A. Mañas	Polytechnic University of Madrid, Spain
Catherine Meadows	NRL, USA

Chris Mitchell	RHBNC, University of London, UK
Refik Molva	Eurécom, France
Eiji Okamoto	University of Tsukuba, Japan
Rolf Oppliger	eSecurity, Switzerland
George Pangalos	Aristotelean University of Thessaloniki, Greece
Ahmed Patel	University College Dublin, Ireland
Guenther Pernul	University of Regensburg, Germany
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Gerald Quirchmayr	University of South Australia, Australia
Jean-Jacques Quisquater	UCL, Belgium
Peter Ryan	University of Newcastle, UK
Kouichi Sakurai	Kyushu University, Japan
Pierangela Samarati	University of Milan, Italy
Sean Smith	Dartmouth College, USA
Diomidis Spinellis	Athens University of Economics and Business, Greece
Julien P. Stern	Cryptolog, France
Michael Szydlo	RSA Security Inc., USA
Moti Yung	Columbia University, USA
Jianying Zhou	Institute for Infocomm Research, Singapore

External Reviewers

George Aggelis	University of the Aegean, Greece
Carlos Aguilar	LAAS-CNRS, France
Walid Bagga	Institut Eurecom, France
Thodoris Balopoulos	University of the Aegean, Greece
Phil Brooke	University of Plymouth, UK
Jeremy Bryans	University of Newcastle-upon-Tyne, UK
Oscar Canovas	University of Murcia, Spain
Shiping Chen	George Mason University, USA
Lazaros Gymnopoulos	University of the Aegean, Greece
DongGuk Han	Kyushu University, Japan
John Iliadis	University of the Aegean, Greece
Kenji Imamoto	Kyushu University, Japan
George Kambourakis	University of the Aegean, Greece
Satoshi Koga	Kyushu University, Japan
Gregorio Martinez	University of Murcia, Spain
Gabriel López Millán	University of Murcia, Spain
Lilian Mitrou	University of the Aegean, Greece
Björn Muschall	University of Regensburg, Germany
Melek Onen	Institut Eurecom, France

Akira Otsuka	Information Technology Promotion Agency, Japan
Thea Peacock	University of Newcastle-upon-Tyne, UK
Josep Pegueroles	Technical University of Catalonia, Spain
Agapios Platis	University of the Aegean, Greece
Fabien Pouget	Institut Eurecom, France
Torsten Priebe	University of Regensburg, Germany
Thomas Quillinan	University College Corke, Ireland
Junji Shikata	Yokohama National University, Japan
BHan Shin	Tokyo University, Japan
Seong Han Shin	Tokyo University, Japan
Vasileios Vlachos	Athens University of Economics and Business, Greece
Chao Yao	George Mason University, USA
Alec Yasinsac	Florida State University, USA
Rui Zhang	Tokyo University, Japan
Sencun Zhu	George Mason University, USA

Table of Contents

Introduction to the Belgian EID Card	1
<i>D. De Cock, K. Wouters, and B. Preneel</i>	
The EuroPKI Experience	14
<i>A. Lioy, M. Marian, N. Moltchanova, and M. Pala</i>	
CERVANTES – A Certificate Validation Test-Bed	28
<i>J.L. Muñoz, J. Forné, O. Esparza, and M. Soriano</i>	
Flexible and Scalable Public Key Security for SSH	43
<i>Y. Ali and S. Smith</i>	
What Is Possible with Identity Based Cryptography for PKIs and What Still Must Be Improved	57
<i>B. Libert and J.-J. Quisquater</i>	
Identity-Based Cryptography in Public Key Management	71
<i>D.H. Yum and P.J. Lee</i>	
Pre-production Methods of a Response to Certificates with the Common Status – Design and Theoretical Evaluation.....	85
<i>S. Koga, J.-C. Ryou, and K. Sakurai</i>	
Filling the Gap between Requirements Engineering and Public Key/Trust Management Infrastructures	98
<i>P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone</i>	
A Framework for Evaluating the Usability and the Utility of PKI-enabled Applications	112
<i>T. Straub and H. Baier</i>	
Using LDAP Directories for Management of PKI Processes	126
<i>V. Karatsiolis, M. Lippert, and A. Wiesmaier</i>	
Recursive Certificate Structures for X.509 Systems.....	135
<i>S. Russell</i>	
A Probabilistic Model for Evaluating the Operational Cost of PKI-based Financial Transactions	149
<i>A. Platis, C. Lambrinoudakis, and A. Leros</i>	

A Practical Approach of X.509 Attribute Certificate Framework as Support to Obtain Privilege Delegation.....	160
<i>J.A. Montenegro and F. Moya</i>	
TACAR: a Simple and Fast Way for Building Trust among PKIs	173
<i>D.R. Lopez, C. Malagon, and L. Florio</i>	
On the Synergy Between Certificate Verification Trees and PayTree-like Micropayments.....	180
<i>J. Domingo-Ferrer</i>	
A Socially Inspired Reputation Model	191
<i>N. Mezzetti</i>	
Using EMV Cards for Single Sign-On	205
<i>A. Pashalidis and C.J. Mitchell</i>	
Distributing Security-Mediated PKI	218
<i>G. Vanrenen and S. Smith</i>	
Distributed CA-based PKI for Mobile Ad Hoc Networks Using Elliptic Curve Cryptography	232
<i>C. Zouridaki, B.L. Mark, K. Gaj, and R.K. Thomas</i>	
ÆTHER: an Authorization Management Architecture for Ubiquitous Computing.....	246
<i>P.G. Argyroudis and D. O'Mahony</i>	
Trustworthy Accounting for Wireless LAN Sharing Communities.....	260
<i>E.C. Efsthathiou and G.C. Polyzos</i>	
Mobile Qualified Electronic Signatures and Certification on Demand	274
<i>H. Rossnagel</i>	
Performance Evaluation of Certificate Based Authentication in Integrated Emerging 3G and Wi-Fi Networks.....	287
<i>G. Kambourakis, A. Rouskas, and D. Gritzalis</i>	
A Credential Conversion Service for SAML-based Scenarios	297
<i>Ó. Cánovas, G. López, and A.F. Gómez-Skarmeta</i>	
A New Design of Privilege Management Infrastructure with Binding Signature Semantics.....	306
<i>K. Bicakci and N. Baykal</i>	
How to Qualify Electronic Signatures and Time Stamps	314
<i>D. Hühnlein</i>	

An Efficient Revocation Scheme for Stateless Receivers.....	322
<i>Y.H. Hwang, C.H. Kim, and P.J. Lee</i>	
On the Use of Weber Polynomials in Elliptic Curve Cryptography	335
<i>E. Konstantinou, Y.C. Stamatiou, and C. Zaroliagis</i>	
Threshold Password-Based Authentication Using Bilinear Pairings	350
<i>S. Lee, K. Han, S.-k. Kang, K. Kim, and S.R. Ine</i>	
An Efficient Group Key Management Scheme for Secure Multicast with Multimedia Applications	364
<i>C.N. Zhang and Z. Li</i>	
Author Index	379

Introduction to the Belgian EID Card

BELPIC

Danny De Cock*, Karel Wouters*, and Bart Preneel

Katholieke Universiteit Leuven,
Department of Electrical Engineering–ESAT SCD/COSIC
Kasteelpark Arenberg 10, B-3001 Heverlee-Leuven, Belgium
<http://www.esat.kuleuven.ac.be/cosic>
{[decockd](mailto:decockd@esat.kuleuven.ac.be),[kwouters](mailto:kwouters@esat.kuleuven.ac.be),[preneel](mailto:preneel@esat.kuleuven.ac.be)}@esat.kuleuven.ac.be

Abstract. This article gives a brief introduction to the Belgian EID card project commonly referred to as “Belpic.” This introduction includes an overview of the history of the project, details on the visual and cryptographic aspects of the EID cards, a discussion of the different sub-CAs involved, together with the card issuing process.

Key words: Electronic Identity card (EID), nation-wide Public-Key Infrastructure (PKI), legally significant certificates, authentication certificates, qualified certificates, Certificate Revocation Lists (CRL).

1 Introduction and Scope

Belgium is planning to be the first European country that distributes Electronic Identity (EID) cards with digital signature technology to *all* its citizens. One of the main incentives to introduce the EID card is to increase the openness of the administration towards the citizen: on the short term, citizens will have access to their own population file to check who was in dialogue with their file during the last months, to trace in what stage is the answer on their request to get a building permit, etc. These are only a few possible uses of the card. Within a few years from now these use cases will have evolved and extended in a variety of new ways, both in the communication channel between the government and its citizens, and between organizations and their customers.

Benefits. All commercial, not-for-profit and governmental players such as banking, insurance, health care, etc. can benefit from the EID card to improve their quality of service without having to implement and deploy their own public-key infrastructure (PKI). They will be able to offer secure remote enrollment, strong entity authentication and digital signatures without the large and expensive overhead of PKI and smart card deployment. Moreover, all entities will

* The author was partially supported by the GOA project MEFISTO 2000/6 of the Flemish Government

have the guarantee from the Belgian government that the citizen/customer has been identified correctly and that the card and its protocols have been evaluated for security. This technology will also significantly decrease the risk for identity theft.

In the short term, the EID card will increase the confidence level of identification information used in the public and commercial sector through a more accurate collection of a citizen's identity data. This will result in an increased use of secure identification and authentication technology in multiple applications, both for governmental and other environments. For many individual applications the issuance of such a card and the establishment of a PKI architecture was not economically feasible, but the EID card allows to reduce this cost.

By January 1st, 2004 all EU member states were required to implement e-invoicing in their national legislation [11]. One condition is that the authenticity of the origin and the integrity of the content is guaranteed, for instance by relying on advanced electronic signatures. It is clear that the current EID card can fulfill these conditions.

Costs and limitations. Today, the citizen pays 12 EUR when he collects his/her EID card, which is about twice as much as for a non-electronic identity card. From the government side, a large investment in infrastructure and management is required in the move to the electronic version.

The current EID card only authenticates the identity of the sender, not his role within the organization or his authorization to perform a certain action (such as sending an invoice) in the first place. This requires application-dependent solutions. Moreover, it has been decided not to support private-key decryption for the pilot phase, in part because of the problem of key recovery for back-up purposes.

One can anticipate that the financial sector (such as the credit card industry, retail payments) will keep issuing its own cards and managing its own PKI architecture. There are several reasons for this: first, they have invested heavily in past years in this environment. To implement a worldwide deployment (e.g., EMV technology in the credit cards); second, they prefer to control their own environment both in terms of technology, risk management and marketing (logos on the card). Finally, multi-application smart cards with secure separation between the applications have not yet reached the required maturity level. One can however expect that the EID card will be used as a bootstrap mechanism for retail e-banking.

The current EID certificates include the name and RRN number (National Register number) of the holder; this is an 11-digit number consisting of the date of birth (dd/mm/yy format), 3 digits reflecting a sequence counter and 2 check digits. This number should be considered as sensitive personal data. Unfortunately, the design of the RRN number makes it rather trivial to guess. The RRN number is used by the government (and by some health care organizations) as a link to the identity of the card holder, because it is the key used as input to many databases containing information about the citizen. The inclusion of the name and RRN number of the citizen in the certificate attached to every transaction

is very convenient for the government, but is clearly not desirable from a privacy point of view. The Austrian “Bürgerkarte” [6] has an improved solution, where a unique identifier is provided per application. In the future one could imagine that advanced private credentials would be used [4, 3].

While the government has well-established and reliable procedures to identify citizens, issuing electronic cards imposes additional requirements on the personnel in the municipalities; any weakness in the issuance process can result in substantial problems later on.

Organization of this article. The remainder of this article presents the history of the project in Section 2. Sections 3 and 4 describe the card and its issuance in detail. Finally, Section 5 presents the Belpic CA hierarchy. Table 1 lists the abbreviations used in this article.

Table 1. List of abbreviations

ARL:	Authority Revocation List
BELPIC:	Belgian ELectronic Personal Identification Card
CA:	Certification Authority
CRL:	Certificate Revocation List
EID:	Electronic IDentity
NOF:	Number of
OCSP:	Online Certificate Status Protocol
PIN:	Personal Identification Number
PKI:	Public-Key Infrastructure
PUK:	Personal Unblocking Key
RRN:	Rijksregister/Registre National – the National Register

2 History of the Belgian EID Card Project

The legal framework for the electronic identity card was initially offered by the EU directive [10] transposed into Belgian law in July 2001 [8]. The Belgian Council of Ministers decided on 22 November 2000 to provide citizens access to e-Government by public-key cryptology and certificates, where the cryptographic keys and the corresponding certificates are stored on a smart card (the Belpic project). A new council decision confirmed the guidelines proposed by a working group on the technical requirements for the card: the chip, the card readers and the certificate practice statements. It was decided to deploy the EID card nation-wide only after a successful pilot tested in eleven municipalities; in this pilot phase 60 000 EID cards are distributed.

Pilot phase. A call for tender was issued for a pilot phase, which consists of the infrastructure development for the National Register and for eleven test municipalities to become operational. This infrastructure development started in 2002;

the software prototype was tested at the National Register and subsequently installed in the eleven municipalities. The first cards were issued in the Spring of 2003 (the first municipality started on 9 May 2003, the eleventh on 25 July 2003). It is planned to renew all non-electronic identity cards within a five years period once the pilot phase has ended. It has been decided that as of the end of 2004, all Belgian citizens will have the possibility to request an EID card, even if they are not living in one of the pilot municipalities, and that by the end of 2007, the majority of all identity cards of Belgian citizens must have been replaced by electronic ones.

Figure 1 shows the number of EID cards that have been issued so far: at the end of March 2004, slightly more than 50 000 EID cards have been produced, and more than 42 000 cards have been issued to citizens. The certificates of approximately 5 500 of these cards have been revoked: about 10% of all the citizens who have received a new EID card have decided they do not want to use the card's digital signing capabilities. This means that so far more than 36 000 EID cards are still active and can be used in e-Government and e-business transactions. Reference [5] provides up-to-date versions of this and many more Belpic-related graphs.

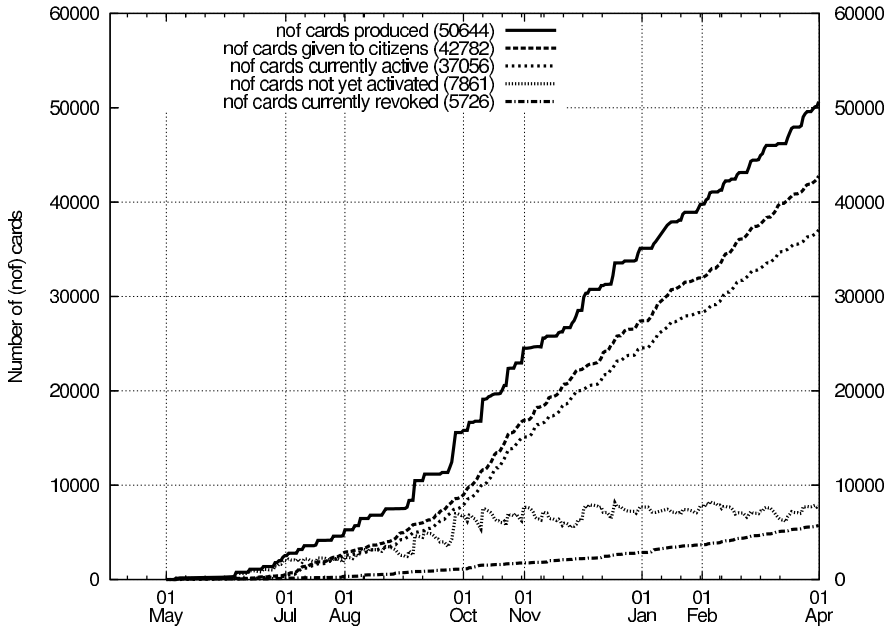


Fig. 1. Current deployment of the EID card. The order of the lines in the legend corresponds to the order of the individual curves

The chip also contains two private keys with a certificate and one private key without a certificate. Furthermore, authentic copies of the certificates of the Citizen Certification Authority (CA), the government's Root CA, and the National Register's certificate are stored on the card. The card holder can use these reference certificates to decide whether to trust the topmost certificate of a certificate validation chain.

Information on the different Certification Authorities is provided in Section 5. The EID card also contains information necessary to update and manage the digital information stored therein.

Digital Signatures The integrity of all the citizen-related digital information in the smart card (JPEG photo, main address, name, etc.) is protected using a digital signature issued by the National Register.

Three Private Keys The EID card contains one card-specific and two citizen-specific private keys. Each of these RSA 1024-bit key pairs is generated on the card during the initialization of the EID card by the Card Initializer (cf. Step 5 in Figure 3). The length of the corresponding public-key exponents can vary between 17 and 256 bits, and is specified by the Card Initializer.

Basic private key – Card authentication. The only card-specific private key, the so-called basic private key, is used to authenticate the EID card towards the National Register. At this stage, only the National Register is able to validate the digital signatures computed with this private key as only the National Register knows which EID card corresponds to which basic public key.

This basic private key is used in the mutual authentication protocol which is executed, e.g., when the citizen-related content must be updated in the EID card, if the citizen moves from one residence to another, or if the government has updated one of its certificates.

Legally significant signing key – Citizen identification. The EID card is a secure signature creation device [10] which can be used to produce digital signatures that are equivalent to hand-written signatures. Once the digital signature has been created by the signer, the signer cannot deny that this signature has been produced by his/her EID card, provided that the signer's certificate was valid at the time when the signature was created. The private signing key used for this type of signature is also known as a non-repudiation signing key. The verification key that corresponds to this private key is linked to the citizen via a qualified certificate. The term "qualified certificate" is used by the European Directive on Electronic Signature [10] to refer to a specific type of certificate, with application in European electronic signature legislation. This certificate's primary purpose is to identify a person with a high level of assurance, where the certificate meets specific qualification requirements defined by an applicable legal framework such as defined in Belgium's implementation of the Directive.

Authentication signing key – Citizen authentication. A citizen can produce a digital signature with the authentication key to provide strong authentication of the citizen for client authentication processes, e.g., for electronic banking, to log onto a web site for e-business or e-Government applications, etc. The corresponding public key is linked to the citizen via an ordinary non-qualified certificate.

4 EID Card Issuing Process

When a citizen has to receive an EID card (e.g., to replace an expired identity card), a complex process is triggered in which the National Register plays a central role. The National Register is a department of the federal government where all the identification information on the Belgian citizens is managed and monitored. All changes in the status of the EID card's production process are reflected in the National Register's databases. The municipalities act as an interface between the National Register and the citizens, and as a Registration Authority for the Citizen CA. The visual information of the citizens' EID card is printed by the Card Personalizer, while the card's digital content is initialized and generated by the Card Initializer. In practice, a single party acts as the Card Personalizer and the Card Initializer.

The complete issuing procedure is depicted in Figure 3. If the citizen spontaneously asks for a new EID card, Step 0 is omitted:

- Step 0:** The citizen receives a convocation letter with an invitation to obtain a new EID card.
- Step 1:** The citizen visits the municipality with his/her picture.
- Step 2:** A civil servant validates the identity of the citizen based on the old ID card of the citizen, and starts the EID card production. The citizen then manually signs the official EID request form. The Card Personalizer will print this hand-written signature on the new EID card.
- Steps 3, 4:** The National Register receives the EID request form from the citizen's municipality and forwards it to the Card Personalizer.
- Step 5:** The Card Personalizer prints a new EID card and has the Card Initializer trigger the generation of the three key pairs on the citizen's EID card (cf. Section 3.2)
- Step 6:** The National Register is informed of the existence of a newly printed EID card which holds the three freshly generated private keys, and it receives the National Register's part of the card's activation code.
- Step 7:** For the legally significant and authentication key pairs, a certificate request is sent to the CA.
- Step 8:** When the CA has issued these two certificates, it publishes their serial numbers on the Certificate Revocation List using the X.509v3 extension "certificateOnHold" [7]. This means that these certificates will be considered as "invalid" if an application checked their validity status.
- Step 9:** The Card Initializer receives the citizen's certificates and stores them on the appropriate EID card.

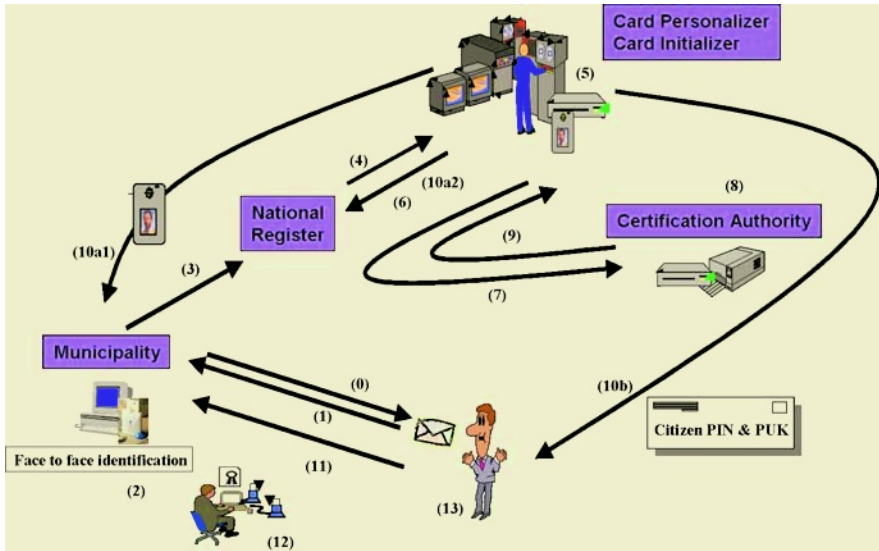


Fig. 3. Belpic EID card production process

Step 10a1, 10a2: When all the digital information has been stored in the EID card (citizen's identity information, address, certificates, etc.), the Card Initializer de-activates the card and sends it to the municipality of the citizen, after which the National Register is informed of this fact.

Step 10b: After the new EID card has left the Card Initializer's premises, the Card Initializer sends a letter with the EID card holder's Personal Identification Number (PIN) and activation code (PUK) to the citizen.

Step 11: When the citizen has received this letter, he/she can visit the municipality to activate and collect his/her EID card.

Step 12: A civil servant fetches the citizen's EID card, verifies the citizen's identity and starts the card's activation process (This requires the civil servant to log onto the National Register's server to access the Register's part of the activation code): the EID card is activated after the successful presentation of the activation code.

Step 13: When the EID card has been activated, the civil servant witnesses that the citizen generates a digital signature with each signing key. If the digital signatures can be verified with the public keys certified in these certificates, the status of the two certificates for this citizen changes from "certificateOnHold" to "active." In practice, this means that the National Register commands the CA to remove the CRL items that correspond to these certificates as soon as the EID card has been activated.

The citizen now has an active EID card, which can be used to generate legally significant and authentication signatures.

5 Belpic CA Hierarchy

The Belpic certificate hierarchy has three levels. The first level is the Belgium Root CA; the second level contains the EID operational CAs, while the third level concerns the certificate users (i.e., the citizens, the operators entitled to manipulate the EID card and the various e-Government entities). All the citizen and e-Government certificates are formatted according to the X.509v3 standard [7]. The EID card administration certificates are formatted according to the role certificates specified in ISO/IEC 7816-4 and 7816-8 [1, 2].

Three distinct Belpic Certification Authorities have been established: the Card Administration CA, the Citizen CA and the e-Government CA. The authenticity of these CAs is protected via two Root CA certificates: a self-signed Root CA which is stored in the EID cards and another Belgium Root CA certificate which can be validated by the main browsers as it is issued by GlobalSign, one of the large commercial CAs of which the Root certificate is embedded in the current releases of the main commercial and non-commercial browsers such as Netscape Communicator, Microsoft Internet Explorer, Mozilla and Opera. This Belgium Root CA certificate is sent to the browser (or any other application that uses strong authentication of the application or web server) when the client would visit a government web site as part of the server's certificate chain.

Figure 4 summarizes the Belpic CA hierarchy. Each of these CAs issues both full CRLs and Delta CRLs. A Delta CRL is a special CRL which enumerates all modifications made to the current full CRL with respect to a previous full CRL, the so-called Base CRL.

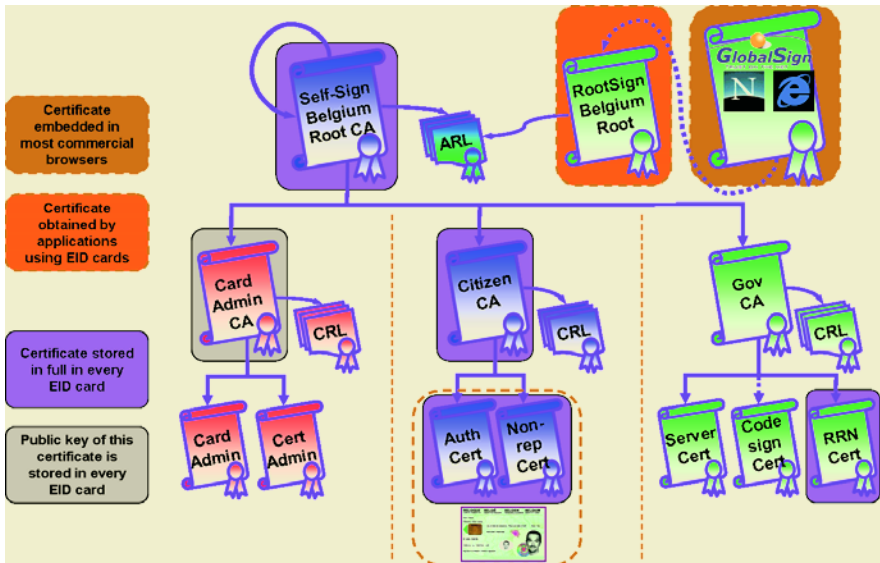


Fig. 4. Belpic EID CA structure

5.1 Card Administration CA

The Card Administration CA issues role certificates [2] to the various parties that are involved in the EID card administration, e.g., government departments that require special access to the EID card. These role certificates are used in a challenge response protocol between the EID card and the holder of the role certificate to authenticate the external party to the EID card. The following table summarizes the actions that are allowed after successful authentication with a role certificate (one role certificate can list one or more roles):

	Roles							
	1	2	3	4	5	6	7	8
update Card Administration CA certificate								*
update citizen's ID information							*	
update reference info of the Municipality						*		
update Root CA certificate					*			
store citizen certificates	*	*		*				
generate new key pairs	*	*	*					
create new directories	*	*						
delete directories	*							
delete key pairs	*							
delete certificates	*							

A role certificate issued by the Card Administration CA can grant one or more roles to the certificate holder, e.g., Role 5 and 8, or only Role 7. For example, if a citizen moves from one place to another, the EID card will only accept the updated address if the update command comes from a party which can produce a digital signature that corresponds to a role certificate for Role 7. The EID card authenticates itself towards that party with the card's basic private key (cf. Section 3.2).

5.2 e-Government CA

The functions taken care of by the e-Government CA are very similar to those of the Citizen CA which is discussed below. In addition to the issuance of certificates for the government's (web)servers, it also supports object signing, e.g., to authenticate the lists of EID card related software and hardware which the government recommends or approves. The digital signatures computed by the National Register on the identity information of the citizen (cf. Section 3.2) can be verified using the National Register certificate which belongs to this CA branch.

5.3 Citizen CA

The major tasks of the Citizen CA consist of the issuance, suspension, activation and revocation of citizen certificates.

Certificate Issuance The Citizen CA issues certificates on the request of the National Register. Each certificate which is issued is immediately suspended at creation, and will only be activated when the citizen presents him/herself at the municipality (cf. step 13 of Figure 3). When the CA receives a valid certificate request from the Card Initializer for a new EID card, the serial number of the corresponding certificate is marked as valid, but not yet activated. In X.509 terminology, this corresponds to the critical extension “certificateOnHold.” Only when the citizen activates his/her EID card at the municipality, this extension is removed from the CRL, which means that if one checked the certificate status after the activation (as long as the citizen does not start the suspension/revocation procedure), the certificate will be valid. Whenever a certificate is activated, the Delta CRL is extended with a “removeFromCRL”-item for the corresponding certificate serial number.

Suspension of Certificates The Citizen CA suspends certificates at the request of the National Register. A citizen can decide to have the certificates of his/her EID card suspended at all times.

Revocation of Suspended Certificates All certificates that have been suspended for over a week will automatically and irreversibly be revoked. Their status changes from “Suspended” to “Revoked.”

Certificate Validation Mechanisms Two certificate validation mechanisms are provided by the Citizen CA: Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol. The Citizen CA issues CRLs in accordance with RFC 3280 [7], including Delta-CRLs. Next to the classic Certificate Revocation Lists, the Citizen CA also provides an OCSP responder which produces OCSP responses conform RFC 2560 [9]. If the validity period of a certificate or a revoked certificate has expired, it does no longer make sense to include it in the CRL: the validity period has expired. Therefore, all serial numbers of expired certificates are removed from the CRLs.

Availability. All CRLs and Delta-CRLs can be collected from <http://crl.eid.belgium.be>. This web site is publicly available using HTTP and HTTPS protocols, and includes information about the plain CRLs and Delta-CRLs, including possible splitting mechanisms which may be introduced should the individual CRLs grow too large.

Issuance frequency. Every three hours, a new full CRL and Delta CRL is issued by the Citizen CA. The Base CRL of a Delta-CRL is never older than 14 calendar days.

6 Summary and Open Problems

This article briefly presents the history and incentives to introduce the EID card in Belgium. It is clear that this card can be used for many applications in which strong authentication of the citizen is required (e.g., e-invoicing, access control, authorization after authentication, etc.), without a significant investment of the commercial, not-for-profit and governmental players as they can reuse the infrastructure setup by the government.

It is remarkable that very few design and implementation decisions had to be fine tuned along the way from the initial design of the Belpic architecture layout and procedures to the layout and procedures that are currently used: the EID cards issued in the pilot phase only support the RSA signature scheme specified in PKCS#1v1.5, where the cards issued after the pilot phase will support the RSA signature schemes specified in PKCS#1v2.1 [12]. The post-pilot EID cards will also support RSA-OAEP decryption [12]. These two advanced signature and decryption schemes had already been included in the initial design specifications, but had been deferred to the post-pilot phase due to practical reasons.

Issues that have not been covered in detail in this article include the privacy aspect of personal information that is stored in the EID card (e.g., birth date and place, home address) and in the citizen's two certificates (e.g., National Register Number). These aspects will be dealt with in the future versions of the EID card.

References

- [1] ISO JTC 1/SC 17. ISO/IEC 7816-4 – Identification Cards, Part 4: Inter-Industry Commands for Interchange. <http://www.iso.org>.
- [2] ISO JTC 1/SC 17. ISO/IEC 7816-8 – Identification Cards, Part 8: Security-related inter-industry commands. <http://www.iso.org>.
- [3] Stefan A. Brands. *Rethinking public key infrastructures and digital certificates: building in privacy*. MIT Press, Cambridge, MA, USA, 2001.
- [4] David Chaum. Showing credentials without identification: transferring signature between unconditionally unlinkable pseudonyms. In J. Seberry and J. Peprzyk, editors, *Advances in Cryptology – AUSCRYPT '90*, volume 453 of *Lecture Notes in Computer Science*, pages 246–264, Sidney, NSW, Australia, 1990. Springer-Verlag, Berlin Germany.
- [5] Danny De Cock. Inofficial information on the Belgian ELection Personal Identification Card (BEPIC). <https://www.cosic.esat.kuleuven.ac.be/belpic/>.
- [6] Stabsstelle IKT-Strategie des Bundes. Die Österreichische Bürgerkarte. <http://www.buergerkarte.at>.
- [7] R. Housley, W. Ford, W. Polk, and D. Solo. RFC 3280 – X.509 Internet PKI Certificate and Certificate Revocation List (CRL) Profile, April 2002.
- [8] Ministerie van Economische Zaken. Wet houdende vaststelling van bepaalde regels in verband met het juridisch kader voor elektronische handtekeningen en certificatie-diensten. Available at <http://www.just.fgov.be/>, accepted 9 July 2001, published 29 September 2001. (In Dutch and French).
- [9] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. RFC 2560 – X.509 Internet PKI Online Certificate Status Protocol - OCSP, June 1999.

- [10] European Parliament. Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community Framework for Electronic Signatures. L 013:0012–0020, January 2000.
- [11] European Parliament. Council Directive 2001/115/EC of 20 December 2001 amending Directive 77/388/EEC with a view to simplifying, modernising and harmonising the conditions laid down for invoicing in respect of value added tax. *Official Journal*, L 015:0024–0028, January 2002.
- [12] RSA Laboratories. PKCS#1v2.1: RSA Cryptography Standard. Available at <http://www.rsasecurity.com/rsalabs/index.html>, June 2002.

The EuroPKI Experience

Antonio Lioy, Marius Marian, Natalia Moltchanova, and Massimiliano Pala

Politecnico di Torino, Dip. di Automatica e Informatica
Corso Duca degli Abruzzi 24, Torino (Italy)
`security@polito.it`

Abstract. This paper discusses the technical and management experience gained in the day-by-day operation of the EuroPKI infrastructure. First the context where EuroPKI was born is explained, along with its certification philosophy. Then common certification practices are discussed, along with description of the services and applications offered by the EuroPKI partners. User-reported problems are also listed and discussed in order to identify the issues that hamper large scale adoption of public-key certificates. The article closes with an overview of the EuroPKI activity plans and perspective.

1 Introduction

EuroPKI is a spontaneous aggregation of partners that believe in the value of public-key certificates and their use for network, application and document security. EuroPKI has its roots in the ICE-TEL and ICE-CAR projects, funded by the European Commission (EC) to promote the development of PKI-based European security technology to protect open networks and advanced network services (such as e-government, e-commerce or e-healthcare).

In July 1996 the ICE-TEL project began operation of its experimental PKI in 10 European countries. This is probably the first example of a running transnational European PKI. Initially, it was based on X.509v1 certificates that were later replaced by X.509v3 certificates to overcome the well known limitations of the v1 format (e.g. lack of expressivity, poor CRL management capabilities). The ICE-CAR project continued on this track, with more emphasis on support for real-life applications.

In January 2000 the ICE-CAR partners decided to create EuroPKI in order to broaden the scope of the infrastructure and to lay the foundation for its autonomous life beyond the end of the project. At the same time, the management of the Root Certification Authority (initially run by UNI-C in Denmark) was assigned to the Politecnico di Torino, who offered to maintain it at least until 2010.

After the end of ICE-CAR, EuroPKI has provided services to other national and international initiatives. Among them, the NASTEC EC-funded project used EuroPKI to promote secure applications in the newly associated states (NAS) to the European Union. Within this frame, two new national CAs were set up in Romania and Poland.

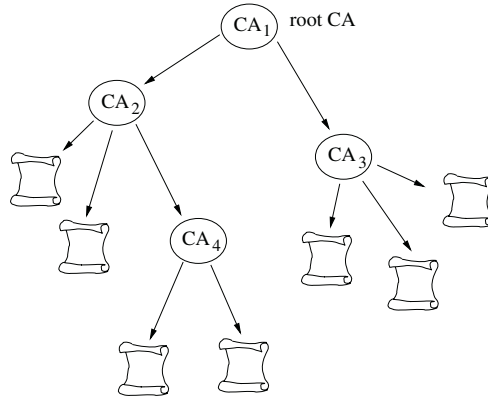


Fig. 1. Hierarchical trust model.

2 The EuroPKI Certification Model

The organization of a PKI reflects the trust model required by its constituency. Three primary models are used: hierarchical, cross-certification, and bridge.

Hierarchical trust (Figure 1) is the most common PKI model: trust is established as a tree structure that flows from top to bottom. At the top of the hierarchy is the root or top-level CA (TLCA): it directly certifies subordinate CAs that in turn provide certification services to their users or to other sub-CAs. This model permits to delegate trust together with CA operations to subordinate authorities. Also, the path construction procedure is very simple (a single path exists from any end entity up to the TLCA). The main disadvantages are the presence of a single point of failure (i.e. the TLCA) and the political issues related to subordination between parties.

In the cross-certification model (Figure 2), there is no subordination: two CAs cross-certify each other if they mutually agree to trust each other's certificates as if they had been issued by themselves. In this model there is no single point of failure, and the CAs are able to act fairly autonomously without being bound by policy delegated from a parent CA. Although many vendors have im-

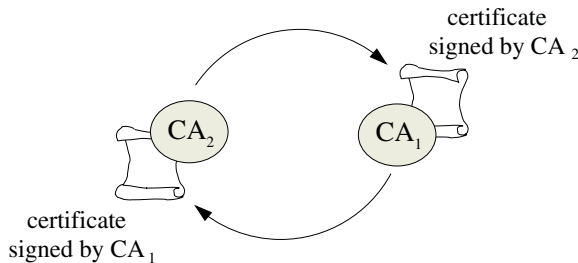


Fig. 2. Cross-certification trust model.

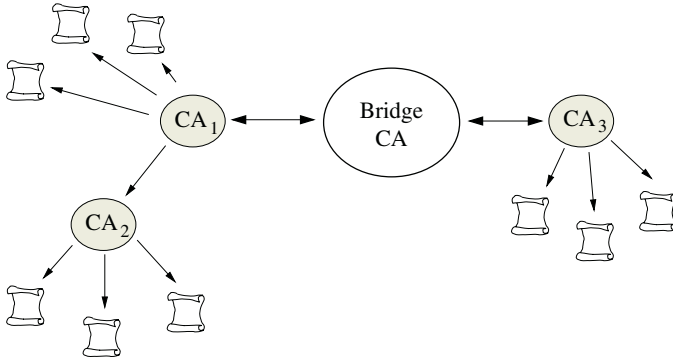


Fig. 3. Bridge trust model.

plemented cross-certification in their PKI management products and the IETF has included CA cross-certification in its Certificate Management Protocol [1], cross-certification is still not well supported by real-life applications.

The Bridge trust model (Figure 3) is similar to the cross-certification one in that there is no single Root CA [2] rather there is one CA - the Bridge CA (BCA) - that acts as a facilitator to interconnect other CAs. A BCA typically does not issue certificates to end entities, but it is used as a hub to interconnect the spokes that can be individual CAs, hierarchical or cross-certified PKIs. With this model, each member needs only to maintain a single cross-certification with the BCA and then it is automatically able to build certification paths across all spokes.

Trust is built into the applications by embedding the certificates of the trust anchors. Usually this takes the form of a *trust list*, that is the list of all CAs directly trusted by the application. Other CAs - in order to be trusted - must have a relation to one CA of the list and the relation type (i.e. hierarchical, cross-certification, bridge) must be understood and managed by the application. Trust lists are of common use in major off-the-shelf applications and provide a very simple solution to the trust management problem for the average user, but they are criticized because the criteria for insertion of a CA in the list are often based more on a commercial rather than a security analysis.

As EuroPKI wants to provide support for every-day user operations, a hierarchical trust model was adopted because this is the only model widely supported by currently available applications. Moreover, it requires only one operation (i.e. addition of the TLCA) to embed trust in the end-user applications, given that the partners didn't want to pay the fees needed to enter commercial agreements with the major application providers. At March 2004, Politecnico di Torino hosts the EuroPKI TLCA. As the number of affiliated national CAs varied since the start of the project, the TLCA has issued so far 18 certificates for 8 different national CAs.

One important feature of EuroPKI is the guarantee of a common ground where all partner operate according to a pre-established set of rules. These rules are specified in the EuroPKI Certification Policy (CP) [3] that establishes CA management rules and the applicability of issued certificates. A well-written CP may help end-users in deciding whether a certificate, and the binding therein, is sufficiently trustworthy for a specific purpose.

The first version of the EuroPKI Certification Policy (CP) was published in October 2000. The policy follows pedantically, section-by-section, the structure of RFC-2527 [4] to allow an easy comparison with this standard. Where a specific section of this RFC was not applicable to EuroPKI, an explicit note was made in the document, but that particular section was not skipped.

The EuroPKI policy successfully stimulated the partners to address issues in their own policies before submitting their versions to the hierarchy. Moreover, the EuroPKI policy was used as a starting point by other organisations, including even some not affiliated to EuroPKI (e.g. the CINECA Grid), when writing their own policy.

3 Common Practices in EuroPKI

An internal survey was conducted at the end of 2003 to better understand the common practices followed by the partners. Gathered data are reported and commented here.

3.1 Certificate Data and Status Distribution

Every partner CA make certificates available via HTTP (100%) and the majority supports also LDAP (54%). Every CA (100%) generates CRL that are made available via HTTP (100%) and sometimes also via LDAP (28%). No FTP or SCEP support was reported for certificate and CRL distribution. 45% of the partners support OCSP. No use of delta CRL [5] was reported, probably due to the small number of certificates revoked so far.

3.2 Certificate and CRL Profile

The survey evaluated also the type and number of extensions used within certificate profiles. The results are summarised in Table 1. The certificate profiles are very similar to each other. The *CRL Distribution Point* extension is widely adopted while the *Authority Information Access* is reported to be used only by some organisations. This is reasonable since not all CAs support the OCSP service.

As will be explained in more detail in section 5.1, a wide adoption of the *Authority Information Access* and *Subject Information Access* extensions should be encouraged as it could help the certificate validation process and the retrieval of certificates.

Table 1. Certificate extensions usage.

Extension Name	Usage (%)
CRL Distribution Point (CDP)	73
Certificate Policy Identifier	91
Authority Information Access (AIA)	54
Subject Alternative Name	91

The survey compared also CRL profiles in terms of CRL version and frequency of publication. It turned out that all CAs publish CRLs at least once a month and almost every CA (82%) uses the v1 CRL format. This is due to the need to keep compatibility with old software (e.g. Netscape Communicator) that does not support the v2 CRL format.

3.3 CA Management Major Issues

CA management is not an easy task, therefore many issues were reported concerning this subject. Three major problem areas have been identified:

- certificate profile management complexity and lack of flexibility
- character encoding support
- lack of needed features in CA management software

The first problem could probably explain why the certificate profiles are so similar across different organisations. In fact, poor understanding of certificate extensions could be the main cause for profile similarity.

3.4 User Related Issues

Users have reported several issues that fall mainly in two broad categories: PKI understanding and mobility problems.

The first class of issues is tied to the lack of knowledge about PKI fundamentals and digital signature technology. As the user interfaces of PKI-enabled applications aren't simple enough and foolproof, lack of PKI knowledge creates a usability problem for the average user. This is not a problem that can be solved by the certificate provider, but rather a call for better certificate handling in applications. Within EuroPKI, several partners have run seminars and prepared user-level documentation to help in using certificates within common applications, but a lot of work in this area is still needed.

The second class of problems is related to user mobility. Users often need to use digital certificates on different workstations and no easy solution is supported by existing applications. Moreover, cryptographic hardware devices (e.g. smart cards or USB tokens) are not yet widely deployed because specific drivers and hooks are needed, both at the OS and application levels, due to scarce support for standard-based cross-platform solutions such as *PKCS#11* [8].

Nevertheless, possible workarounds do exist. For example, applications could be configured to retrieve user credentials and trust anchors from a *PKCS#12* formatted file [7] that might be stored on a removable device like a floppy disk or a USB memory token. As all the user data is carried within this file (i.e. user's certificate, private key and other PKI data), no installation of additional software or extra configuration would be required. Also, the adoption of a more standardised approach for the integration of crypto-devices should be considered by applications and crypto-hardware vendors since this would help users in avoiding security problems (e.g. malicious code seeking for memory-stored crypto keys) and key-recovery issues (e.g. after a system crash).

3.5 Most Valuable PKI-enabled Applications

EuroPKI partners find that S/MIME secure mail is the most popular application. The second place is shared by access authentication (e.g. via SSL/TLS client authentication) and electronic document signature and encryption.

4 EuroPKI Services and Applications

Since the real value of a PKI is not in the certificate itself but in its application, several EuroPKI partners have developed certificate-based applications and provided application-oriented services, such as on-line certificate status verification (via OCSP) and Time Stamping (via TSP). To simplify the enrolment procedures, Registration Authorities have been studied and deployed. Moreover, different PKI-based applications have found a good test ground in this framework; an example of a successful one, WebSign, is described in section 4.4.

4.1 The Centralised EuroPKI OCSP Responder

The *online certificate status protocol* (OCSP) - defined in RFC-2560 [6] - is used to check on-line the status of a single certificate, so avoiding the burden associated to CRL download and analysis. TORSEC¹ developed back in the year 2000 an OCSP responder to be used within EuroPKI.

The policy behind the EuroPKI hierarchy favoured the scenario with a centralised OCSP responder that provides its service on behalf of all the CAs in the hierarchy. To this aim, the TLCA issues to the centralised OCSP responder a special certificate that contains two particular extensions:

- ***extendedKeyUsage*** with the *id-kp-OCSPSigning* object identifier. In this way the TLCA delegates the authority of signing OCSP responses to the owner of this certificate.
- ***id-pkix-ocsp-nocheck*** with a NULL value to state that the OCSP responder's certificate can be trusted across its entire validity period. The usage of this extension suggests to constrain the lifetime of the certificate: as this

¹ The security group of Politecnico di Torino - <http://security.polito.it>

extension allows the client to skip validation of the responder's certificate, to minimise the risks related to key compromises and their effects on the PKI relying parties, the lifetime of the server's certificate was restricted to a maximum of 4 months.

With this certificate profile, the OCSP server acts as a *delegated* responder for the EuroPKI TLCA, and as a *trusted* responder for all the other CAs present in the hierarchy. In literature, this mixture of the two traditional OCSP trust models is sometimes called the *family* model.

The EuroPKI OCSP responder uses CRLs as its source of revocation information, to preserve consistency between statuses obtained via CRL and OCSP. To always have the most accurate and up-to-date revocation information, the responder periodically looks for a new CRL on each CA in the hierarchy and eventually downloads it. CA administrators can also explicitly trigger CRL retrieval whenever a new CRL is issued. Each time a new CRL is downloaded, the OCSP responder performs a restart to load the new revocation data.

To avoid replay attacks, the EuroPKI responder is not using pre-produced answers: it always signs the OCSP responses on the fly. However this poses performance problems. The main bottleneck singled out during the server's development and testing is represented by the cryptographic capabilities of the hosting system's CPU. The throughput of the responder is always upper bounded by the maximum number of signatures per second that the hosting system is able to perform. Consequently, a considerable part of the development effort was devoted to minimise the overhead of the OCSP protocol itself. The tests run on various hardware platforms demonstrate that the server's performance is within 80-90% (in terms of requests per seconds served) of the maximum cryptographic performance of the server's CPU (in terms of signatures per seconds made).

4.2 The Registration Authority Service

Under the EuroPKI policy an out-of-band ID authentication is required before a certification request is approved and the corresponding certificate is issued.

In most PKI environments, the request authentication may be performed by a Registration Authority (RA). There can be several scenarios. For example one organisation may outsource the certification service from an external vendor but still internally perform the RA function. Another scenario could involve the presence of many RAs disseminated over a wide area to maintain close contacts with the users. In all these cases, the CA would require proper authentication of the RA operator before issuing a certificate.

There are different solutions to the problem of RA operator authentication. For example, the OpenCA project implements web-based signed forms to approve users' request via common browsers (i.e. Internet Explorer, Mozilla or Netscape Communicator). This approach uses existing technologies - such as Javascript and SSL/TLS client authentication - to perform on-line RA operations. This approach has some limits, though. First of all, web-based form signing

is not supported by every browser and anyway it is not standard, so that different scripts are required depending on the target browser. Additionally, all the RA operations must be performed on-line.

To avoid these limitations, TORSEC developed an RA tool to connect securely to the CA front end and perform operations request approval or rejection. This software, based on the OpenSSL crypto library, has a graphic Win32 client for the RA operator and a Unix gateway to the CA management software. Operator identification is possible via software or hardware cryptographic credentials: the client supports *PKCS#12* files as well as smart-cards and USB crypto tokens via a *PKCS#11* interface to the OpenSSL Engine extension.

The RA operator connects to the server over an SSL channel that requires certificate-based client authentication. Access to a specific queue of pending certification requests is then granted by checking the operator identity against a signed Access Control List (ACL). Multiple ACLs can be stored on the same server and they can be related to one or more CA. This allows the RA server to be accessed by RAs affiliated to different CAs at the same time. Once authenticated, the operator can download the pending requests, sign the approved ones and send them back to the RA server. The requests are then queued on the server and wait to be exported to the corresponding CA for certificate issuing. As different RAs can access the same server, multiple operators can exist in different locations thus allowing third parties to act as Registration Authorities on behalf of one CA. A schema of the RA data flow is shown in Figure 4.

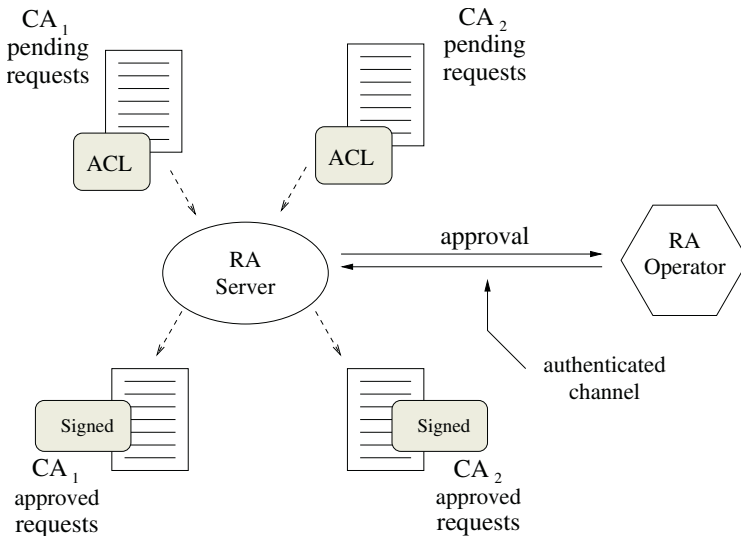


Fig. 4. RA client-server data flow.

4.3 The EuroPKI Time Stamping Service

A time stamp is an electronic seal that includes a time indication. In practice it's a digital signature over a submitted digest, a time indication and other information. The RFC-3161 [9] defines the Time Stamping Protocol (TSP) to be used for requests and responses between the entities involved: the requester or client, and the Time Stamping Authority (TSA) or server.

Based on software developed by the TORSEC group, EuroPKI has established a TSA connected to a NTP stratum-1 server. The TSA server is compliant with RFC-3161 but supports only raw TCP access. For security reasons, client access can be restricted by accepting only signed requests or SSL connections with client authentication.

The TSA certificate has some peculiarities. Unlike the OCSP responder, there is no problem with key compromise: if it happens, the TSA certificate can be revoked and hence no special restriction must be posed to the certificate lifetime. However, to be used for time-stamping, the certificate contains the *extended-KeyUsage* extension with only the *timeStamping* value, marked as critical.

4.4 WebSign, a Successful PKI-based Application

WebSign is a client-server platform for on-line document preparation, submission, signing, storage and processing. This platform was developed by SETCCE² for use in various business, academic and governmental scenarios, where a transition from paper based records to electronic documents enhances the productivity and reduces the costs. The platform is based on XML documents with XML-signatures and advanced security mechanisms, to provide a high security.

The service has been put into production at SiOL, the biggest Slovenian Internet operator. The platform developed will foster the use of electronic contracts and digital signatures in business environments. Through WebSign, users are able to manage subscription and other Internet provision related services. The WebSign service has been introduced to the public in September 2003 and in the first 6 months the application client has been downloaded from the provider site about 12,000 times. The WebSign users use Qualified Certificates (according to the e-signature European directive) issued by four Slovenian certification service providers.

5 Known Issues

5.1 Repositories and Data Retrieval

CAs are required to publish certificates and CRLs in public repositories. Although this could not seem a serious problem, general and simple availability

² Security Technology Competence Centre - <http://www.setcce.org>; funded by Slovenian Telecom, Jozef Stefan Institute, University of Ljubljana and Infotehna, SETCCE is the administrator of the EuroPKI Slovenian branch

of PKI data is still an unresolved issue. This is related to two different aspects: lack of pointers to the available data and the variety of access protocols.

The problem is very evident when considering people, not being part of a partner organisation, who look for one or more issued certificates. Without focusing on trust related issues, a feasible solution to correctly provide pointers to published data is an extensive usage of the *Authority Information Access* (AIA) and *Subject Information Access* (SIA) extensions. The former can provide information on the issuer of the certificate while the latter carries information (inside CA certificates) about services offered. The *Subject Information Access* extension can carry an URI to point to certificate repositories and Time Stamping services. Hence this extension allows to access services by several different protocols (e.g. http, ftp, ldap or smtp).

Within EuroPKI, public data is mostly available via HTTP. Other protocols (e.g. LDAP) are seldom used or not advertised outside the issuer's organisation. The AIA and SIA extensions are almost never used to point to data sources.

5.2 Multiple E-mail Addresses Support

One of the most valuable PKI applications is protection of e-mail messages. Although RFC-3280 requires specification of the e-mail address in the *subjectAltName* extension, some CAs still encode this information in the Distinguished Name (DN) by using the *Email* field. This has an obvious disadvantage when support for more than a single address in one certificate is required: multiple *Email* fields could raise compatibility issues with widely adopted e-mail clients.

To test application support for multiple e-mail addresses in the same certificate, we issued certificates under two different profiles. In profile A two different e-mail addresses were coded in the *subjectAltName* extension, while in profile B they were inserted by using multiple *Email* fields in the subject DN. Table 2 shows the test results with popular S/MIME e-mail clients. Results confirm that the support for multiple e-mail addresses can be achieved only when using the *subjectAltName* extension. It should be noted that some organisations encode multiple e-mail addresses by using simultaneously the two described methods. Although this should be avoided, it is reported to work sometimes.

Table 2. E-mail client tests.

E-Mail Client	Profile A	Profile B
Netscape 4.7	Supported	Not Supported
Netscape 6	Supported	Not Supported
Netscape 7.1	Supported	Not Supported
Mozilla 1.5	Supported	Not Supported
Thunderbird 0.6	Supported	Not Supported
Microsoft Outlook 6	Not Supported	Not Supported

5.3 Non US-ASCII Character Sets

When eastern European countries entered EuroPKI, the problem of non US-ASCII characters in certificates had to be faced. In RFC-2277 [10] Alvestrand suggests the need to clarify whether the strings used in a protocol are subject to internationalisation or not. Although this RFC does not mandate for a policy on name internationalisation, it requires that all protocols be able to use the UTF-8 character set. This consists of the ISO 10646 [11] character set combined with the UTF-8 [12] character encoding scheme. The RFC-3280 promotes the adoption of the UTF8String encoding type to allow for special characters to be properly stored: conforming CA's must encode attributes of DirectoryString type as UTF-8 after December 31, 2003. While this can be more or less easily achieved when issuing certificates, the real problem lies on the application side. Several software packages still do not support the UTF8String type properly. We list here our findings on the behaviour of widely used applications.

Open issues are present in some versions of MS Windows. When a certificate is displayed by an application running under Windows-NT4, Windows98 or Windows95 and the subject field of the certificate contains UTF-8 characters, the data is not correctly visualised. According to Microsoft Knowledge Base Article 824197 the problem source is in the operating system itself and no fix is planned. Hence full UTF-8 support for X.509 certificates is currently available only in Windows ME, Windows 2000, Windows XP, and Windows Server 2003.

The Netscape Communicator suite (i.e. Netscape Navigator and Netscape Mail) up to version 4.x - still in use in some environments - does not support UTF-8 encoded strings. This causes the application not to even import user certificates containing this kind of data. It is therefore impossible to use such certificates with this suite.

The Mozilla and Firebird browsers had a bug which produced some unreadable characters when displaying certificate details. This was due to the fact that non US-ASCII characters (encoded as BMPString) were displayed by using iso-8859-1 encoding. Although it was not possible to read the special characters, no other function was affected. Although a patch for this bug (#185167) is available at <http://bugzilla.mozilla.org>, it has not yet made its way to the stable distribution at April 2004.

The Opera suite version 7.23 correctly supports certificates with UTF-8 encoded strings both in certificate visualisation and usage. However Opera uses certificates only for SSL since it doesn't support S/MIME in its mail client.

5.4 Certificate Renewal

The EuroPKI policy permits to renew an expired certificate. It is therefore possible to issue a new certificate for the same public key, with no need to regenerate a new key pair. However this creates a problem with Internet Explorer that cannot deal with this situation: the new certificate simply doesn't show up in the certificate management interface.

A solution to this problem is to import the renewed certificate together with the corresponding private key; this can be done by using a PKCS#12 file where these data are bundled together. It seems that IE needs, for each imported certificate, a corresponding private key, thus being not able to bind more than one certificate to a single private key.

5.5 Naming Rules in Distinguished Names

The EuroPKI policy does not impose specific rules about the format of the Distinguished Name beside the need for names to be meaningful. However, due to law or administrative requirements this could not always be true.

As an example let us consider the case of one affiliated Italian CA that needed a special format in the Common Name (CN) for compliance with the Italian digital signature law [13]. The requested format is:

$$CN = \langle \textit{surname} \rangle / \langle \textit{name} \rangle / \langle \textit{personal_tax_id} \rangle / \langle \textit{unique_id} \rangle$$

This format of the CN carries data meaningful only to applications compliant with the Italian digital signature law. Moreover the DN must contain another non optional field: the *Description* one. It is composed by several sub-fields that carry data able to confuse a standard X.509 parser even more:

$$Description = "C = \langle \textit{surname} \rangle / N = \langle \textit{name} \rangle / D = \langle \textit{birth_date} \rangle / [R = \langle \textit{role} \rangle]"$$

The presence of slashes or commas can raise compatibility issues because these characters are mostly used as fields separators and some parsers could misinterpret these contents. This is an example of bad usage of certificates that break compatibility across applications.

To improve interoperability and lifetime of certificates, it would be better that specific data, where absolutely needed, be encoded by using the *otherName* type in the *subjectAltName* extension. To certify roles, the adoption of Attributes Certificates should also be considered.

6 Future Plans

Besides maintaining and expanding EuroPKI, our future efforts are aimed to develop better PKI support for applications and systems.

A first area to be addressed is better certificate processing in complex cases. Deep hierarchies, bridge CA, multiple sources of revocation status (CRL, OCSP, indirect CRL, ...) require careful definition of procedures when building the certificate path up to a trusted root and verifying the status of all the certificates in the chain. An exact algorithm needs to be defined and implemented as a library to support applications.

Proper configuration of thousands of workstations to support PKI is a nightmare for a security administrator. In the same way, implementing PKI support into light devices (such as PDAs or handheld PCs) with reduced networking and

processing capabilities is very difficult. These two problems can be simultaneously solved by the concept of “lightweight PKI” where activities that are heavy or require centralised configuration, are delegated to a trusted PKI server. Along this line, the IETF PKIX working group has already defined the requirements for delegating the validation process to a dedicated server. Various protocols, including OCSP with extensions, SCVP [14] and DVCS [15] were proposed for communication among the client and the delegated server. We have already started to investigate the proposed protocols and our efforts are directed towards implementing the servers and integrating support for certificate validation in light devices (e.g. we are currently working with Windows CE handheld PCs).

On the PKI management side, our efforts are directed to support bulk registration and renewal. Large universities and public administrations with tens of thousands of users face severe problems related to management complexity and registration services costs. For example, let us consider a University that needs to issue 30,000 certificates: if the registration process requires five minutes for each certificate (an optimistic value!) the whole process would require 150,000 minutes or more than 14 man/month! And this is without counting the time needed to solve technical problems or correct human errors. Therefore research and development of new tools to support bulk registration and approval is going on, to introduce a certain degree of automation in the certification process. Our current approach is to perform automatic data retrieval from existing data bases and to re-engineer the standard registration services.

7 Conclusions

Through its four years history, EuroPKI demonstrated its usefulness in supporting the security needs of end-users. Additionally, it has helped partner organizations to better understand the technical, policy and management issues involved in setting up and maintaining an operative PKI.

Experience has shown that the problems on the certification authority side can always be solved: technical standards do exist, products implement them and most of the times it is just a matter of proper configuration and management.

The real problems lie on the application and end-user sides. Application developers have not security as a priority, have limited understanding of PKI and are not well supported by transparent and complete libraries.

8 Acknowledgements

The authors want to thank the people and organizations that contributed to creating and developing the EuroPKI infrastructure: the current partners (CSP, Ezitrust, IAIK, NASK, Politecnico di Torino, Politechnika Lodzka, Politehnica University of Bucharest, Provincia di Macerata, PWR Centre for Networking and Supercomputing, SETCCE, University of Modena and Reggio Emilia) as well as the past ones and every partner of the ICE-TEL, ICE-CAR and NASTEC projects.

The authors also thank the European Community for the support provided by funding the mentioned research projects, SECUDE³ for providing free of charge its CA management suite to the EuroPKI TLCA, and the open-source community for many tools that we use in our everyday operations.

A special thank to those people that back in the '90s have introduced us to the wonderful field of PKI: David Chadwick, Borka Jerman Blazic, Steve Kille, Peter Kirstein, Sead Muftic and Wolfgang Schneider.

References

- [1] C.Adams, S.Farrell, "Internet X.509 Public Key Infrastructure Certificate Management Protocols", RFC-2510, March 1999
- [2] W.T.Polk, N.E.Hastings, "Bridge Certification Authorities: Connecting B2B Public Key Infrastructures", NIST, September 2000
- [3] "EuroPKI Certificate Policy - Version 1.1", EuroPKI website, <http://www.europki.org>
- [4] S.Chokhani, W.Ford, "Certificate Policy and Certification Practices Framework", RFC-2527, March 1999
- [5] R.Housley, W.Polk, W.Ford, D.Solo, "Certificate and Certificate Revocation List (CRL) Profile", RFC-3280, 2002
- [6] M.Myers, R.Ankney, A.Malpani, S.Galperin, C.Adams, "Online Certificate Status Protocol - OCSP", RFC-2560, June 1999
- [7] RSA Laboratories: "PKCS#12: Personal Information Exchange Syntax Standard", Version 1.0, June 24, 1999
- [8] RSA Laboratories: "PKCS#11: Conformance Profile Specification", Version 2.11, October 1, 2000
- [9] C.Adams, P.Cain, D.Pinkas, R.Zuccherato: "Time-Stamp Protocol (TSP)", RFC-3161, August 2001
- [10] H.Alvestrand: "IETF Policy on Character Sets and Languages", RFC-2277, January 1998
- [11] ISO/IEC: "Information Technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane", May 1993, with amendments
- [12] F.Yergeau: "UTF-8, a transformation format of ISO 10646", RFC-2279, January 1998
- [13] AIPA: "CIRCOLARE 19 giugno 2000 n. AIPA/CR/24", Italian MIT Website, <http://www.innovazione.gov.it>, 2000
- [14] A.Malpani, R.Housley, T.Freeman: "Simple Certificate Validation Protocol (SCVP)", IETF Draft, PKIX Working Group, October 2003
- [15] C.Adams, P.Sylvester, M.Zolotarev, R.Zuccherato: "Data Validation and Certification Server Protocols", RFC-3039, February 2001
- [16] C.Weider, C.Preston, K.Simonsen, H.Alvestrand, R.Atkinson, M.Crispin, and P.Svanberg: "The Report of the IAB Character Set Workshop held 29 February - 1 March, 1996", RFC-2130, April 1997

³ <http://www.secude.com>

CERVANTES – A Certificate Validation Test-Bed^{*}

Jose L. Muñoz, Jordi Forné, Oscar Esparza, and Miguel Soriano

Technical University of Catalonia (Telematics Engineering Department)
1-3 Jordi Girona, C3 08034 Barcelona (Spain)
{jose.munoz,jordi.forne,oscar.esparza,soriano}@entel.upc.es

Abstract. Certificate validation is one of the toughest scalability problems of the PKI. The goal of this paper is to introduce a Java platform for certificate revocation called CERVANTES. CERVANTES pretends to be an easy to extend tool that allows researchers to develop and test their own “real” revocation systems. As CERVANTES is an open source project it can also be included as part of any open PKI project. The platform is very flexible and due to its modular design it allows for example, to fit a new kind of status checking protocol without having to recompile the source code. CERVANTES includes our implementations of the main standards (CRLs and OCSP) as well as an implementation of a system based on the Merkle Hash Tree (one of the most popular systems among the non-standard ones). Finally, we use CERVANTES to obtain performance results about each developed system. These results guarantee that CERVANTES runs as expected.

1 Introduction

The Public Key Infrastructure (PKI) is responsible for the Identity Certificates (ICs) not only at the issuing time but also during the whole life-time of the certificate. An IC has a bounded life-time: it is not valid prior to the activation date and it is not valid beyond the expiration date. In this context, certificate revocation can be defined as “*the mechanism under which an issuer can invalidate the binding between an identity and a public-key before the expiration of the corresponding certificate*”. Thus, the existence of a certificate is a necessary but not sufficient evidence for its validity, the PKI needs to provide applications that use certificates with the ability to check, at the time of usage, that the certificate is still valid¹. Figure 1 presents the reference model that we use to describe the certificate revocation paradigm. Regarding the PKIX model [5], our reference model removes what it is not directly involved in the revocation and it depicts the entities and the mechanisms that are directly related to the revocation process in more detail.

^{*} This work has been supported by the Spanish Research Council under the project ARPA (TIC2003-08184-C02-02) and the European Research Council under the project UBISEC (IST-FP6 506926).

¹ Actually, the validation of a certificate comprises another mechanism: the certification path validation, but our platform currently does not address this feature.

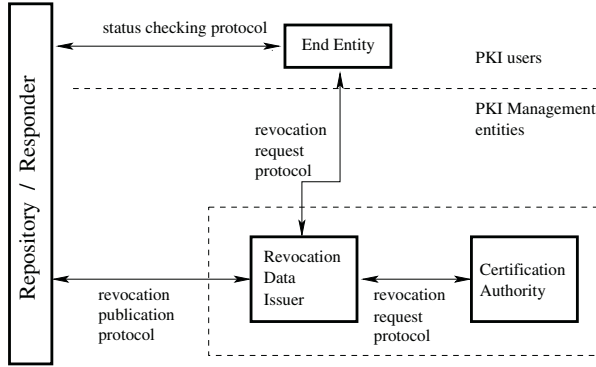


Fig. 1. Revocation Reference Model

The revocation process starts with a request to revoke a certain certificate. Usually, the owner of the certificate to be revoked, an authorized representative or the issuer CA, can create revocation requests, but, in general, a Certification Practice Statement (CPS) should define who is able to perform this task in each PKI domain. To revoke the certificate, one of the authorized entities generates a revocation request and sends it to the Revocation Data Issuer (RDI). RDI² is the term that we use to define the Trusted Third Party (TTP) that has the master database of revoked certificates. The RDI is also responsible for transforming the revocation records from the database into “status data”. The status data has the appropriate format in order to be distributed to the End Entities (EEs). The status data usually includes the following fields: *Certificate Issuer* (the Distinguished Name or DN of the CA that issued the target certificate or certificates), *Validity Period* (this period of time bounds the status data life-time and it is obviously smaller than the validity period of the certificate), *Issuer Name* (the DN of the TTP that issued the status data), *Cryptographic Evidence* (the evidence must demonstrate that the status data was issued by a TTP), *Serial Number* (the serial number of the target certificate or certificates), *Revocation Date* (the date when the target certificate was revoked), and optionally the *Reason Code* (a revocation reason for guidance can be specified). In the vast majority of the revocation systems, EEs do not have a straight connection to the RDI. Instead of this, the RDI publishes the status data in “repositories” or “responders”. The main function of both repositories and responders is to answer the EE requests concerning the status of certificates (status checking). The difference between them is that repositories are non-TTPs that store status data pre-computed by the RDI while responders are TTPs that have a private-key and that provide a signature (that serves as cryptographic evidence) for each response. It must be stressed that status checking is the most resource-consuming mechanism in

² The CA that issued the certificate is often the one that performs the RDI functions for the certificate, but these functions can be delegated to an independent TTP.

the overall revocation process and the one that carries the major part of the interoperability problems.

This paper introduces CERVANTES, a Java platform for certificate revocation. CERVANTES pretends to be an easy to extend tool that helps researchers to develop and test their own real revocation systems. As CERVANTES is an open source software project, it can also be included as part of any open PKI project. One of the main design goals for us was to make CERVANTES easy to extend. For this reason, CERVANTES has a modular design and well-defined APIs between each of its modules. The final platform is very flexible and due to this modular design and it allows for example, to fit a new kind of status checking protocol without having to recompile the source code.

Currently, CERVANTES includes our implementations of the main standards (CRLs [16] and OCSP [14]) as well as an implementation of a system based on the Merkle Hash Tree [13] (one of the most popular approaches among the non-standard systems). In this paper, we use CERVANTES to obtain some performance results about each developed system. These results permit us to check that the platform behaves as expected. However, as a reviewer noticed, the goal of CERVANTES is not to provide results that can be obtained with simple pen-and-paper calculations³. CERVANTES pretends to be a tool to perform tests in a wide sense, including real implementations, protocols and interoperability issues⁴. The basic JAVA structure of CERVANTES was published in [9] and more evaluation results with CERVANTES can be found in [11]. This paper deals with the logical organization of the platform, including the functions of each module and the interrelations between modules. The rest of the paper is organized as follows: in Section 2 we present the main approaches to certificate status checking. In Section 3 we present CERVANTES (Certificate Validation Test-bed). In Section 4 we validate the behaviour of CERVANTES for the main status checking mechanisms. Finally, we conclude in Section 5.

2 Related Work

There are many approaches that define the data format to be exchanged with the End Entities to perform the status checking. The simplest of them is the traditional *Certificate Revocation List* (CRL) [6]. CRL is the most mature approach and it has been part of X.509 since its first version. CRL has also been profiled for the Internet in [5]. A CRL is a digitally signed list of revoked certificates in which for each entry within the list the following information is stored: the certificate serial number, the revocation reason and the revocation date. The CRL has

³ In this sense, some models have been proposed to evaluate PKI components in general and the certificate revocation in particular. For instance, in [10] we presented an analytical way of modeling revocation, while in [3], Arnes presents a simulation model. However, in our opinion a model or a simulator can not address all the aspects of a real system and it might overlook important details.

⁴ Although we have performed some interoperability tests with other systems, these tests are not presented here due to the room limitation.

also a header that includes information about the version, the CRL serial number, the issuer, the algorithm used to sign, the signature, the issuing date, the expiration date and some optional fields called extensions. The CA that issued the certificate acts as RDI and repositories can be used to distribute the CRLs. Since CRLs may have a large size, they are usually cached by the client during their validity period. *Overissued* CRL (O-CRL) [4] addresses a way of reducing the peak request rate of CRLs towards the repositories and evenly distribute the requests across time. O-CRL simply consists in issuing more than just one CRL during a validity period. The result will be that the CRLs in relying parties' caches will expire at different times, so requests to the repository for new CRLs will be more spread out. *Delta-CRL* (D-CRL) [6] is an attempt to reduce the size of the CRLs. A Delta-CRL is a small CRL that provides information about the certificates whose status have changed since the issuance of a complete list called Base-CRL. *CRL-Distribution Points* (CRL-DP) was introduced in the version 3 of X.509 [6]. In CRL-DP, each CRL contains the status information of a certain subgroup of certificates. Each subgroup is associated with a CRL distribution point, which can be located on the same or different repositories. Each certificate has a pointer to the location of its CRL distribution point, so there is no need to either search through distribution points or have a priori knowledge of the revocation information locations. The criteria for creating these subgroups can be geographic, by their level of importance, scope of use, etc.

The *Online Certificate Status Protocol* (OCSP) [14] has been proposed by the PKIX workgroup of the IETF. In OCSP the status of certificates is available in responders through a request/response mechanism. An OCSP client issues a status request for a particular certificate and sends it to an OCSP responder. The acceptance of the certificate in question is suspended until the responder provides a response. Upon receipt of a request, the responder determines whether the request is correct, searches the status information in its local database (which can be a CRL), creates a response with the corresponding data, signs this response and sends it back to the client.

The *Certificate Revocation Tree* (CRT) [7] and the *Authenticated Dictionary* (AD) [15] are both based on the Merkle Hash Tree (MHT) [8]. The MHT relies on the properties of the OWHF (One Way Hash Functions). It exploits the fact that a OWHF is at least 10,000 times faster to compute than a digital signature, so the majority of the cryptographic operations performed in the revocation system are hash functions instead of digital signatures. In [13] we present the details of a revocation system called AD-MHT, which is based on the MHT and the data structures proposed in [15].

3 CERVANTES

CERVANTES⁵ is based on a client/server architecture where the clients are the EEs that perform the status checking and the revocations and the server

⁵ The home of CERVANTES is located at <http://isg.upc.es/cervantes>

performs the functions of the RDI, the responders and the repositories. The software is organized in three parts: the Client, the Server and the Library. Below, we describe them.

3.1 The Library

The Library implements the functions that are common to the clients and the server and it is necessary to run all of them. Among the Library functions the most important is to provide the ASN.1 support for both clients and server. As CERVANTES is built in Java, we need to convert the ASN.1 PDU definitions into Java classes. In order to perform this conversion, the ASN.1 definition file of the PDU is used as input to an ASN.1-to-Java stub compiler. In particular, we use *Snacc4Java v2.3* [1]. In computer security, ASN.1 data are normally encoded using the Distinguished Encoding Rules (DER) [17] because these rules give exactly one way to represent any ASN.1 value as a bit-pattern. Thus, DER is appropriate for applications in which a unique bit-pattern encoding is needed, as is the case when a digital signature is computed on an ASN.1 value. In our case, the *Snacc4Java* libraries let us perform the DER encoding of the ASN.1 objects. DER encoded PDUs can be sent using many operational mechanisms such as raw sockets, HTTP, FTP, LDAP, SMTP, and X.500. So far, CERVANTES only supports raw-sockets and HTTP. However, this already does not imply a great loss of inter-operability because HTTP is the most widely spread transport mechanism among the PKI products. When HTTP is used as operational protocol, the HTTP header contains the **Content-Type** set to the proper MIME type and the **Content-Length** set to the total length in bytes (including the header), while the body contains the bit-pattern corresponding to the DER encoded PDU.

3.2 The Server

The server is the most complex part of CERVANTES and it is organized in four modules: the Input/Output (I/O), the Database (DB), the Central Management (CM) and the Status Checking Handlers (SCHs).

The DATABASE MODULE (DB)

This module contains the classes necessary to retrieve, add, delete or modify the information that contains the database. The database is extern to CERVANTES. Any SQL-based database that can be accessed through the JDBC API can be used. In particular, we have chosen *PostgreSQL v7.2.2* because it is very stable and it is an open source project as CERVANTES. The main function of the DB module is to provide an API to perform the operations related to the database. This API defines all the functions that are required by the rest of the modules of the system. These functions permit to retrieve all the records from the database, ask for the number of records that has the database, figure out if a certain record is stored in the database, delete all the records of the database, insert a revocation record (revoke a certificate), and delete a revocation record (remove a record of an expired certificate).

The INPUT/OUTPUT MODULE (I/O)

The I/O module manages the database inputs (revocations) and outputs (expirations). The I/O can be configured to run in two modes:

- *Standard mode.* In this mode CERVANTES can be used as a standard revocation platform, that is, it receives requests from authorized entities to revoke certificates and it responds to requests about the status of certificates. The protocol to revoke a certificate that we use is the Simple Certificate Revocation Protocol (SCRP) [9]. SCRП is a lightweight version of CMP [2] that we have developed. We do not use straight CMP because it agglutinates much functionality that is beyond the needs of a revocation system (and this makes CMP difficult to implement).
- *Test mode.* In this mode, random inputs are used, and CERVANTES provides information that allows us to test the systems without having real users.

From now on, we will focus on the second run mode where the revoked and expired certificates are randomly generated. The random generation process can be controlled by configuring the certificates population (i.e. the number of certificates " N "), the percentage of revoked certificates " r " and the average rate of events " λ_{events} ". When a test is running, two execution threads: the *revocations generator* and the *expirations generator*, are in charge of generating random revocations and random expirations at the correct average rate. The generation of random events is divided in two phases:

(1) *Static phase.* In this phase the *revocations generator* works at a very high rate whereas the *expirations generator* remains in standby. The tests are always started with an empty database and basically the objective of this phase is to fill with revocation records the database until the average number of revoked certificates " n " is reached: $n = N * r$.

(2) *Dynamic phase.* Once the database has reached the average number of revoked certificates, the I/O module reduces the rate of the *revocations generator* and starts the *expirations generator* with a proper rate in order to keep the system dynamic and stable. The objective is insert and delete records in the database at a rate λ_{events} .

Notice that as the target certificate for either a revocation or an expiration is randomly chosen, there are events that do not have any effect. For instance, nothing will happen if the *revocations generator* generates a revoked certificate that is already contained in the database or if the *expirations generator* generates an expiration for a certificate that it is not in the database. It follows from the previous discussion that the average rate of events of the *revocation generator* " $\lambda_{revocations}$ " must be $\lambda_{events}/(1 - r)$. While the average rate of events of the *expirations generator* " $\lambda_{expirations}$ " must be λ_{events}/r . The random events are generated following an exponential probability density function, where the time between events t is obtained by means of a uniform variable " x ", where $x \in [0, 1]$

$$x = 1 - e^{\lambda t} \Rightarrow t = -\frac{1}{\lambda} \ln(1 - x) \quad (1)$$

Once an event is generated, the I/O module sends the information related with it to the CM module.

The CENTRAL MANAGEMENT MODULE (CM)

The CM module is the one that reads the configuration file and initiates the server. The CM also creates a log file in which it will store all the traces of the execution of the server such as the time the server is started, the name of the host, the modules it has found, the running mode etc. Depending on the running mode the CM starts the proper I/O execution threads. The CM also initiates the JDBC connection and creates the main execution threads (PortThreads) of each SCH found in the configuration file.

While the server is running, the CM receives the revocations/expirations from the I/O and it inserts/deletes the records related with the new events in the database. The CM also informs the SCHs about the new events and periodically orders the SCHs to dump and reset their statistic counters. When the server must be stopped, the CM kills all the execution threads.

The STATUS CHECKING HANDLERS (SCHs)

The SCHs are the most complex modules in the platform and they contain the logic necessary to send the status data in the proper format to the EEs. Figure 2 depicts the organization and the transactions that take place among the different elements of a generic SCH.

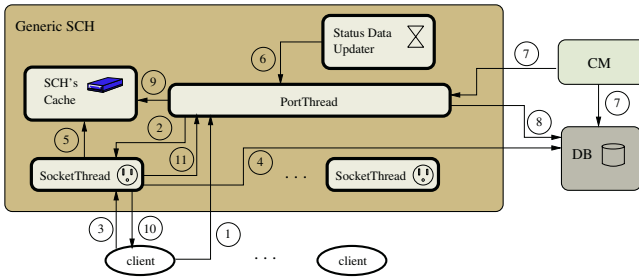


Fig. 2. Generic Status Checking Handler (SCH) organization.

1. Each SCH has a “PortThread” which is an independent execution thread that listens for requests addressed to a certain TCP port.
2. The SCHs are able to serve several requests at the same time, to do so the PortThread creates an independent execution thread for each request called “SocketThread”.
3. The SocketThread receives the data of the request and builds a proper response.

4. The `SocketThread` can obtain the status data required to build the PDU of the response from the database.
5. The `SocketThread` can also obtain the status data required to build the PDU of the response from a local cache.
6. The `StatusDataUpdater` is an execution thread that periodically asks for the `PortThread` to update the cache.
7. The CM informs the database and all the `PortThreads` about each new event (expiration/revocation) that occurs in the system. This fresh information can be used by the `PortThreads` to update their caches.
8. For the `PortThread` another way to update its cache is to access the previously recorded data that is stored in the database.
9. The `PortThread` updates the cache when the `StatusDataUpdater` gives the order to do so.
10. Once the `SocketThread` has all the necessary data (from either the database or the local cache), it sends the response back to the client.
11. Finally the `SocketThread` informs the `PortThread` about the bytes and processing time required to serve the request.

The configuration parameters of a generic SCH are presented in Figure 3.

type of server. This is a string that identifies the protocol of SCH.

class of server. This identifies the class that implements the status server. With this parameter a developer can add new status checking protocols to CERVANTES without having to recompile the code.

name. This is a string that gives a name to the SCH.

operational protocol. This is the underlying protocol that will transport the status data. This can be HTTP, raw or `!auto-detect!`. With the last configuration the server will detect and use the operational protocol used by the client.

port. The number of TCP port used to listen to requests.

server log. This is the path to the file that stores the traces of the execution of the status server for debugging.

statistics log. This is the path to the file that stores the statistics taken by the status server.

certificate path. This is the path to the certificate used to sign the PDUs by SCH.

private key path. This is the path to the file that stores the private key associated with the certificate of the SCH.

Fig. 3. Configuration parameters of a generic SCH server.

Next, we particularize the behaviour of the generic SCH for status checking with CRLs, OCSP and MHT.

CRL-SCH. Our implementation of CRL allows the overissuance of CRLs so besides the validity period "*VP*" of the CRLs, it is necessary to define the overissuance factor "*O*". This means that "*O*" CRLs will be issued during a *VP*.

Figure 4 shows the behaviour of the SCH that manages the CRL

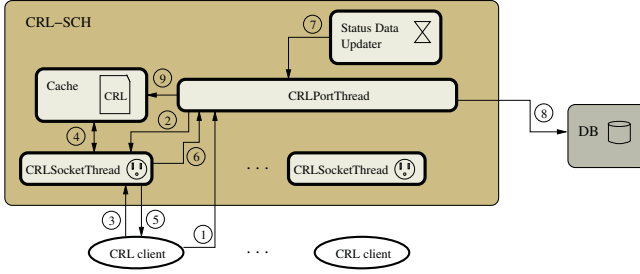


Fig. 4. CRL-SCH

1. The CRLPortThread listens for requests addressed to a certain TCP port.
2. For each request the CRLPortThread creates a CRLSocketThread.
3. The CRLSocketThread receives the request.
4. The CRLSocketThread retrieves the CRL from the cache.
5. The CRLSocketThread sends the CRL to the client.
6. The CRLSocketThread informs the CRLPortThread about the bytes and processing time required to serve the request.
7. Every *VP/O* the StatusDataUpdater asks the PortThread to update the CRL.
8. The CRLPortThread retrieves all the records from the database.
9. With all the status data retrieved, the CRLPortThread generates the list of revoked certificates, sets the validity period, signs the CRL, DER encodes the CRL and sends the result to the Cache.

OCSP-SCH. Figure 5 shows the behaviour of the SCH that manages the OCSP.

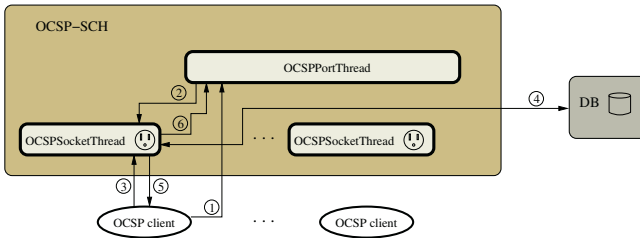


Fig. 5. OCSP-SCH

1. The OCSPPortThread listens for requests addressed to a certain TCP port.
2. For each request the OCSPPortThread creates an OCSPSocketThread.
3. The OCSPSocketThread receives the request.

4. The OCSPSocketThread retrieves the status data for the requested certificate from the DB.
5. With the status data the OCSPSocketThread builds the response including the signature, DER encodes the response and sends it to the client.
6. The OCSPSocketThread informs the OCSPPortThread about the bytes and processing time required to serve the request.

MHT-SCH. Our MHT-based implementation is called AD-MHT [13]. We have also developed an ASN.1 protocol for AD-MHT [12]. Figure 6 shows the behaviour of the SCH that manages AD-MHT.

1. The ADMHTPortThread listens for requests addressed to a certain TCP port.
2. For each request the ADMHTPortThread creates an ADMHTSocketThread.
3. The ADMHTSocketThread receives the request.
4. The Cache stores two MHTs: the “listening tree” and the “management tree”. The listening tree is used by the SCH to respond for status checking requests and it is immutable during the validity period of the *Digest*. The management tree is updated for each expiration/revocation and after a validity period “VP” the management tree is cloned and the listening tree is replaced with the clone. Thus, the ADMHTSocketThread retrieves the status data ($Digest + Path$) from the listening tree stored in Cache.
5. The ADMHTSocketThread sends the response to the client.

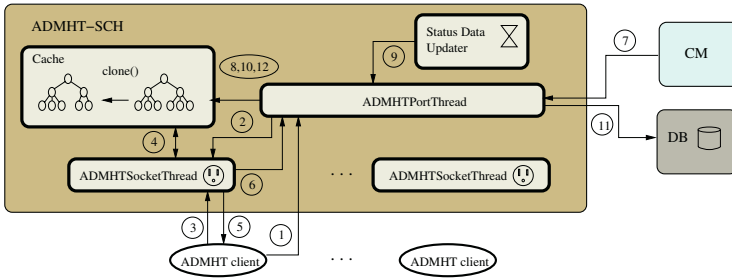


Fig. 6. ADMHT-SCH

6. The ADMHTSocketThread informs the ADMHTPortThread about the bytes and processing time required to serve the request.
7. The CM informs the ADMHTPortThread about any change in the status data.
8. When the ADMHTPortThread is informed about a change in the status data (revocation/expiration) it must update the management tree.

9. Every *VP* the *StatusDataUpdater* asks the *ADMHTPortThread* to clone the management tree.
10. The *ADMHTPortThread* clones the management tree and signs the *Digest* of the new listening tree.
11. If CERVANTES is restarted and it must keep the revocation records from a previous execution, the *ADMHTPortThread* has to build the “first” MHT based on these records. So it retrieves all the recorded status data from the database and builds this first listening tree. If CERVANTES is started with an empty database, the listening tree will also be empty.
12. The *ADMHTSocketThread* sends the first listening tree to the Cache.

3.3 The Clients

The clients are entities able to send requests and process responses using a particular protocol. CERVANTES implements each protocol as a module and each module provides a java API. Any application can use these APIs to perform the necessary client-side operations on behalf of a certificate-using application. Basically the protocol modules send requests with the parameters specified through their API, set a timeout and wait for a response from the server. The communication is closed and a “timeout fail” is reported through the API if the response does not arrive prior to the end of the timeout. If a response is received on time, it is verified and the result is reported through the API. Using the APIs of the different protocol modules, we have developed two applications: a graphical client and a test client. Below, we describe in more detail the test client.

The test client is a multi-threaded application in which each execution thread represents a client that generates random status checking requests. The aim of the test client is to analyze the behaviour of each status checking protocol or a combination of them under different conditions. In order to perform a test we define “groups of clients”. A group of clients is a determinate number of execution threads (clients) with the same configuration. To define a group of clients we need to specify the URL of the server (name/@IP and TCP port) where the requests must be addressed. We also need to define the statistics of the requests. The statistic used is an exponential probability density function in time. In the configuration file you can set the number of clients “ n ” and average rate of requests per hour and client “ $\lambda_{requests}$ ”.

4 Evaluation with Cervantes

A few requirements can be defined to evaluate revocation systems:

Population Size. The absolute size of the number of potentially revocable certificates can strongly influence the approach taken. Obviously, a solution intended to address a large population may require more resources and complexity as compared to a smaller group.

Latency. The degree of timeliness relates to the interval between when a RDI made a revocation record and when it made the information available to the

relying parties. A more eager mechanism to update and convey this information will proportionally consume more bandwidth.

Connectivity. Does the relying party need to be online in order to ascertain the reliability? Online mechanisms create critical components in the overall security design because they make difficult to ensure that the system is operational and functional at any given moment.

Finally, it is also worth mentioning that status checking is the mechanism that has the greatest impact on the overall performance of the certificate revocation system. Therefore, a status checking needs to be fast, efficient and timely, and it must scale well too. It is therefore necessary to minimize two fundamental parameters: the processing capacity and the bandwidth (the amount of data transmitted). In this section we use the CERVANTES platform to evaluate the main status checking mechanisms: CRLs, OCSP and MHT. The experimental results have been obtained under the following conditions: the CERVANTES server runs in a Pentium III with Linux, the clients generate status checking requests per hour following an exponential probability density function, each client has a certificate, there is an average of 10% revocation, the validity period of a CRL is 6 hours, the CRLs are cached by clients during their validity period, clients will request their local cache instead of the repository if they have a cached CRL, and the launching of the test clients is distributed along the first validity period.

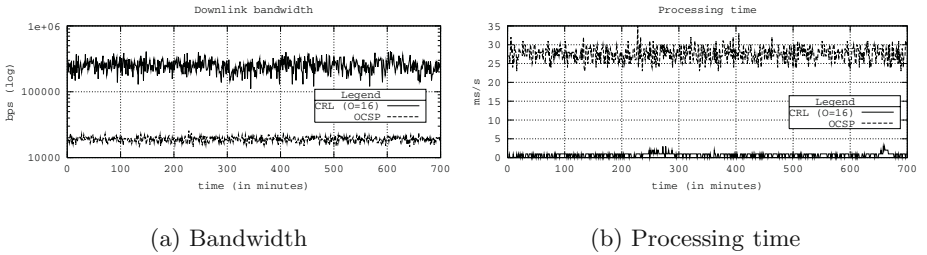


Fig. 7. Temporal evolution of OCSP vs CRL.

Figure 7 shows the temporal behaviour of the Overissued-CRL (with an overissuance factor: $O = 16$) and the OCSP in terms of downlink bandwidth and processing time in the repository/responder (using the configuration previously described). The results show evidence of the bottlenecks of each system: while in the CRL system the figure of the downlink bandwidth is over an order of magnitude bigger than in the OCSP, it happens the contrary in the figure of the processing time.

Figure 8 shows the comparison among O-CRL, OCSP and AD-MHT. Notice that despite the use of cache, the CRL performance in terms of down-link bandwidth is very poor compared to OCSP or AD-MHT. As a result, CRLs

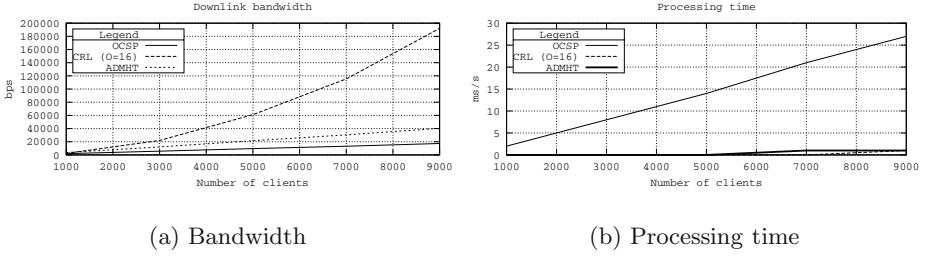


Fig. 8. Scalability regarding the number of clients: AD-MHT versus O-CRL and OCSP

do not seem the best choice for distribution of status data among end users and they should be only used as the distribution mechanism for intermediate entities. OCSP is a good choice in terms of bandwidth but the processing capacity resources it uses are the highest of the evaluated systems. Also, responders are needed in order to distribute the OCSP data. AD-MHT bandwidth performance is slightly worse than that of OCSP, but taking into account the overall performance, the AD-MHT might be a good choice for distribution of status data among end users because it does not require much bandwidth or processing capacity and repositories (non-TTPs) can be used to respond to AD-MHT requests.

5 Conclusions

This paper introduces CERVANTES, a Java platform for certificate revocation. CERVANTES pretends to be an easy to extend tool that allows researchers to develop and test their own real revocation systems. As CERVANTES is an open source project it can also be included as part of any open PKI project. The platform is very flexible: it allows to fit new status checking protocols and new type of tests without significant changes in the structure or the source code. CERVANTES includes our implementations of the main standards (CRLs and OCSP) as well as an implementation of a system based on the Merkle Hash Tree (one of the most popular approaches among the non-standard ones). We have also used CERVANTES to obtain performance results about each developed system. The results obtained permit us to affirm that the platform behaves as expected. It is also worth to mention that CERVANTES is still an open project and we are working into making the software publicly-available.

As future work, it should be interesting to present more results regarding other issues such as usability, management or as a reviewer pointed out, other input patterns rather than random, for example, to study the behaviour of the revocation systems in a situation like a denial of service attack.

It is worth to mention that CERVANTES is still an open project and that more development is currently going on, such as the division of server's program into different programs: one for the revocation data issuer and one for the responder and the repository. Finally, we are also working to include certification path validation in the platform.

References

1. SNACC for JAVA. <http://www.alphaworks.ibm.com/tech/snaccforjava>.
2. C. Adams and S. Farrell. Internet X.509 Public Key Infrastructure Certificate Management Protocols, 1999. RFC 2510.
3. Andre Arnes, Mike Just, Svein J. Knapkog, Steve Lloyd, and Henk Meijer. Selecting revocation solutions for PKI. In *NORDSEC '95*, 1995.
4. D.A. Cooper. A model of certificate revocation. In *Fifteenth Annual Computer Security Applications Conference*, pages 256–264, 1999.
5. R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile, 1999. RFC 2459.
6. ITU/ISO Recommendation. X.509 Information Technology Open Systems Interconnection - The Directory: Autentication Frameworks, 2000. Technical Corrigendum.
7. P.C. Kocher. On certificate revocation and validation. In *International Conference on Financial Cryptography (FC98). Lecture Notes in Computer Science*, number 1465, pages 172–177, February 1998.
8. R.C. Merkle. A certified digital signature. In *Advances in Cryptology (CRYPTO89). Lecture Notes in Computer Science*, number 435, pages 234–246. Springer-Verlag, 1989.
9. J.L. Muñoz and J. Forné. Design of a Certificate Revocation Platform. In *International Conference on Information Technology: Research and Education (ITRE 2003)*. IEEE Communications Society.
10. J.L. Muñoz and J. Forné. Evaluation of Certificate Revocation Policies: OCSP vs. Overissued CRL. In *DEXA Workshops 2002. Workshop on Trust and Privacy in Digital Business (TrustBus02)*, pages 511–515. IEEE Computer Society, September 2002.
11. J.L. Muñoz, J. Forné, O. Esparza, M. Soriano, and D. Jodra. Evaluation of Certificate Revocation Systems with a JAVA Test-Bed. In *DEXA Workshops 2003. Workshop on Trust and Privacy in Digital Business (TrustBus03)*. IEEE Computer Society.
12. J.L. Muñoz, J. Forné, O. Esparza, and M. Soriano. A Certificate Status Checking Protocol for the Authenticated Dictionary. In *Computer Network Security*, volume 2776 of *LNCS*, pages 255–266. Springer-Verlag, September 2003.
13. Jose L. Muñoz, J. Forné, O. Esparza, and M. Soriano. Certificate Revocation System Implementation Based on the Merkle Hash Tree. *International Journal of Information Security (IJIS)*, 2(2):110–124, 2004.
14. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP, 1999. RFC 2560.
15. M. Naor and K. Nissim. Certificate Revocation and Certificate Update. *IEEE Journal on Selected Areas in Communications*, 18(4):561–560, 2000.

16. W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2002. RFC 3280.
17. ITU-T Recommendation X.690. ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), 1995.

Flexible and Scalable Public Key Security for SSH^{*}

Yasir Ali and Sean Smith

Department of Computer Science/PKI Lab
Dartmouth College, Hanover NH 03755 USA
yasir.ali@alum.dartmouth.org
sws@cs.dartmouth.edu

Abstract. A standard tool for secure remote access, the SSH protocol uses public-key cryptography to establish an encrypted and integrity-protected channel with a remote server. However, widely-deployed implementations of the protocol are vulnerable to man-in-the-middle attacks, where an adversary substitutes her public key for the server's. This danger particularly threatens a traveling user Bob borrowing a client machine.

Imposing a traditional X.509 PKI on all SSH servers and clients is neither flexible nor scalable nor (in the foreseeable future) practical. Requiring extensive work or an SSL server at Bob's site is also not practical for many users.

This paper presents our experiences designing and implementing an alternative scheme that solves the public-key security problem in SSH without requiring such an a priori universal trust structure or extensive sysadmin work—although it does require a modified SSH client. (The code is available for public download.)

Keywords: SSH, man-in-the-middle.

1 Introduction

In the UNIX world, users traditionally used `telnet` and `ftp` to access remote machines (to establish login sessions and transfer files, respectively). However, these commands transmitted userids and passwords in the clear, and the increasing insecurity of the networks over which these commands operated have made this risk unacceptable.

Consequently, the *secure shell* (*SSH*) (e.g., [2,10,11,12,13]) has emerged as the de facto replacement for these commands. Rather than typing `telnet` and `ftp` to reach a remote machine *S*, the user invokes `ssh`, which uses public-key cryptography establish authentication and encrypted communications over unsecured channels. The server presents a public key, and the client machine uses standard cryptography to establish a protected channel with the party knowing the private key—presumably, the server. SSH can even permit the user to authenticate via a key pair instead of a password; however, we conjecture that most users stay with the simpler authentication.

^{*} This work was supported in part by the Mellon Foundation, by Internet2/AT&T, and by the Office for Domestic Preparedness, U.S. Department of Homeland Security (2000-DT-CX-K001). The views and conclusions do not necessarily represent those of the sponsors. A preliminary version of this paper appeared as Technical Report TR2003-441, Department of Computer Science, Dartmouth College.

However, common SSH implementations overlook an important property: the binding of the server's public key to the identity of the server to which the user intended to connect. This oversight makes the user susceptible to man-in-the-middle attacks, in which the adversary substitutes her public key for the server's; if the user then authenticates via passwords, the adversary can gain complete control of the user's account.

This risk is particularly pronounced in settings where a traveling user is borrowing a client machine that does not *a priori* have a trusted copy of the intended server's public key. We stress that in this model, the user may trust the client machine he or she is borrowing—but not the client machine's network environment. (Indeed, the second author encountered this: a trusted colleague's machine, in an institute suffering DNS attacks.)

Solving these problems in SSH requires introducing a way for SSH clients to securely bind public keys to servers. Solving these problems in the real world requires that any particular SSH client that any particular user wishes to use be able to perform this binding for any particular server the user might want to connect to.

In the long-term, the DNSSEC vision—using PKI to secure all DNS information—would enable a nice solution (e.g., [7]); however, we don't see this happening in the near-term. Perhaps the next natural approach would be to establish a traditional hierarchical PKI for SSH servers; all SSH clients would know the trust root; all SSH servers would have access to a CA/RA system that would sensibly bind the public keys to usable names; trust paths for any given server would somehow arrive at any given client. (Indeed, this is the approach we first considered; and similar commercial offerings have since emerged.)

However, this natural approach does not meet our real world constraints (Sec. 3.2). This universal trust structure needs to be in place before the traveling user can securely connect from a remote machine. Furthermore, many system environments do not provide a natural hierarchy of certifiers or machine names. (For example., one colleague at a corporation “bar.com” cited 10^4 machines with names of the form foo.bar.com, and whose names were changed apparently at whim by remote sysadmins.)

Alternatively, one might consider a many-rooted trust structure, consisting of many domains linked by bridges and cross-certification. However, it is not reasonable to assume that this solution, attractive in theory, will be workable in wide-scale practice any time soon. (For example, efforts to use bridging to achieve painless interoperability between academic domains academic-to-government domains in the US create ongoing research and engineering challenges.)

Yet another solution might be to build on the “universal PKI” that already exists on desktops: the trust roots built into browsers, and the burgeoning support for browser personal keystores for users. To that end, we considered (and also began prototyping) some additional approaches that used the browser-based SSL PKI to authenticate servers, and possibly clients too. However, this approach would require that all users be affiliated with a site that sets up and maintains an SSL Web server (and pays for annual renewal of a certificate from a standard browser trust root); while assuming a home Web server was reasonable (and common in many academic and corporate environments), we felt that assuming an SSL server was not. This approaches thus loses flexibility.

This consideration left us with the challenge: how do we bring public-key security to SSH, in a way that provides the flexibility and scalability that can permit easy adoption in the real world, without requiring an a priori trust structure or an SSL server?

Sect. 2 provides the background knowledge to understand the problem. Sect. 3 describes the particular risk and usage scenarios we envisioned, and the design constraints that resulted. Sect. 4 presents the solution we developed (in a sense, a decentralized PKI that requires neither certificates nor CAs). Sect. 5 presents architectural and implementation¹ details. Sect. 6 considers some alternate approaches and future work.

2 Background

2.1 The SSH Protocol

First, we consider the SSH protocol (e.g., [2,10,11,12,13]).

When a user on a client machine tries to establish a secure channel with a remote machine, what really happens is the SSH client (on the client host) carries out this protocol with the *SSH daemon* on the server host.

Put simply, SSH allows these two hosts to construct a secure channel for data communication using Diffie-Hellman key exchange, which provides a shared secret key that cannot be determined by either party alone. The shared secret key established is used as a session key. Once an encrypted tunnel is created using this key, the context for negotiated compression algorithm, and encryption algorithm are initialized. These algorithms may use independent keys in each direction. The first session key established is randomly unique for every session.

SSH allows for both the server and the client to authenticate using DSA, as part of this exchange. Typically, the client will authenticate the server by comparing a fingerprint of the server's public key with one stored in a file on the client machine; if they do not match, the user on the client machine would either receive a warning with the option of accepting the new fingerprint as it is, or the client would drop the connection. (We conjecture that the typical user would click "OK" and keep going.)

There are three main parts of the SSH protocol: algorithm negotiation, authentication, and data encryption.

Algorithm negotiation is mainly responsible for determining the encryption algorithms, compression algorithms and the authentication methods supported and to be used between the client and the server. Authentication [13] is further broken down in two pieces: key exchange (transport layer) and user authentication (user authentication layer).

The purpose of the key exchange is dual. Firstly, it attempts to authenticate the server to the client. Secondly, it establishes a shared key which is used as a session key to encrypt all the data being transferred between the two machines. The session key encrypts the payload and a hash generated for integrity checking of the payload using the private key of the server. The client verifies the server's public key, verifies the

¹ Prototype source and binaries are available at

<http://www.cs.dartmouth.edu/~sws/yasir-thesis>.

server signature received and then continues with user authentication. User authentication methods that are supported and are a part of the SSH protocol include passwords, public key, PAM, and Kerberos.

Once authentication is successful, one of the negotiated encryption algorithms is used to encrypt the data transferred between the two machines. Other features that are a part of SSH clients include port forwarding, however such features will not be discussed in this paper.

2.2 Tools

Popular open source tools are the OpenSSH client and server, distributed freely with Red Hat Linux distributions; and the TeraTerm SSH client for Windows. Popular commercial versions include the SSH client and server developed by SSH Inc.

The commercial SSH provides support for a traditional PKI, as we discuss later. However, the OpenSSH client current at the time of our experiments (openssh3.4) did not provide any code that verifies certificates or certificate chains. Rather, certificates were merely treated as public keys. If a key *blob* (the technical name for a data structure that loads the public key) contains a certificate instead of a public key, OpenSSH had routines that can extract the public key out of the certificate and after that it only uses that public key for authentication and integrity checking purposes.

3 Usage Scenarios and Constraints

3.1 Basic Vulnerability

In many public-key applications, the relying party can directly verify that the other entity knows a private key matching a particular public key. However, to draw a more useful conclusion from this, the relying party needs to establish a binding between this public key and some relevant property of this entity.

In the case of SSH, the user needs to establish that the server with whom his client has just established a session is the server to whom he intended to connect.

If the client machine has an *a priori* relationship with the server to which the user wants to connect, under the same server name the user wants to use, and the server's key has not changed, then the user possesses such a binding inductively. (Indeed, many installations suggest that a user traveling with a laptop first ssh to his desired home hosts while safely within a trusted home LAN, in order to pre-load the laptop's fingerprint store.)

However, if these things do not hold, then the user is at risk (e.g., see [9]). When the server sends its public key, the user has no way to verify if this key matches the intended server. It is trivial for an attacker to sit in the middle and intercept the connection, and send her own public key and signature instead. The user may then send his own password to the attacker thinking that she is the intended server.

If the client already has the server's public key, the user will generally receive a warning such that "server's key has changed. Continue?" Most users typically hit "Yes" and do not realize the risk. However, if it is the first time the user is connecting to this

server from this client, the client will not have the server's public key stored locally, and the user will be none the wiser.

(Researchers have also identified how SSH encryption can protect the plaintext content on a channel but still leak other information [8], but we do not consider those issues here.)

3.2 Real-World Constraints

In the trust model we consider, the user trusts the client machine and his intended remote server to work correctly. However, the user does not trust the server machine he actually connects to until he verifies the server's identity; he also does not trust the client's network environment, including its DNS. If the client does not have a trusted fingerprint of the server already loaded, how does the user establish the binding?

We enumerate some desired goals of an effective solution:

- The solution should enable users from borrowed (but trusted) clients, in untrusted network environments, to establish trusted connections to their home machines.
- The solution should be adoptable in the near-term by small groups of users with only a small delta from the current infrastructure.
- The solution should accommodate users in domains where conscientious sysadmins can set up trustable and usable CA services.
- The solution should also accommodate users in domains where no such services exist.
- The solution should *not* require that a new universal PKI structure (neither single-rooted nor multi-rooted) be established before any of this works.
- The solution should *not* require that a user memorize the fingerprints of all servers he wishes to interact with.

3.3 Existing Solutions

As mentioned earlier, a natural approach is to assume the existence of a traditional PKI via which any SSH client can authenticate the binding of the server's public key to its identity. Both SSH Communications Security and F-Secure have had commercial offerings in this area, and new commercial offerings continue to emerge.

However, such approaches did not meet our scalability and flexibility constraints. A universal trust structure must exist before the system is useful; before Bob can use an SSH client at Carla's college, Bob's servers must be certified, and Carla's clients must know a certification path from a trust root to that server. Furthermore, not all domains will have the appropriate personnel and organizational structure to establish reliable and usable bindings in the first place.

We could also require that users carry with them some hardware device (such as a smart card or key fob) that remembers the public keys of the servers they wish to interact with. However, we feel this would be overly awkward and expensive, violating the goals; also, sufficiently robust software and hardware support for such tokens has not permeated the current infrastructure, violating the "small delta" constraint.

4 Our Solution

4.1 Overview

To solve the problem, the client needs to have some trust root upon which to build the conclusion that the binding of the server’s public key to identity is meaningful. The “small delta” and “no new universal PKI” constraints mean that we can neither hard-code one trust root into all clients, nor assume that each client will have a local trust root with a path to Bob’s server. The “no memorization” constraint means that the user cannot bring it with them.

This analysis thus forces us to have the client download the user’s trust root over the network. Since changing how the SSH protocol itself works would also violate the “small delta” constraint, we need to download this data out of band. However, this raises a conundrum: if the user cannot trust the network against man-in-the-middle attacks on the public key the server sends in SSH, how can the user trust the network against man-in-the-middle attacks against this out of band data?

To answer this, we use a *keyed MAC*—a “poor man’s digital signature.” Also known as a keyed hash, a keyed MAC algorithm takes a message M and a secret key k , and produces a MAC value $Mac(M, k)$ with the property that it is believed infeasible to find another M', k' pair that generate the same keyed MAC. Thus, if Bob knows secret key k , and retrieves a message M accompanied by a MAC value h which he confirms as $Mac(M, k)$, then Bob can conclude that M was produced by a party that knew k and that M has not been altered in transit. (In a sense, this like a digital signature, except we have a symmetric key instead of a key pair, and thus we lose non-repudiation.)

Our constraints dictate that we cannot force the user to memorize a public key—but users easily memorize URLs and passphrases. Our general solution thus has two parts. First, we built a *modified SSH client* that accepts a location (such as URL) and passphrase from the user; the client then fetches the the trust root from that URL, and verifies its integrity by deriving a symmetric key from the passphrase and using that to check a keyed MAC on the root data. We then built *configurator* tools to create these MAC’d trust roots in the first place.

4.2 Approaches

We now discuss a range of solutions this basic strategy permits.

Decentralized At one extreme, we imagine a user who wants to securely access his standard machines, but does not have a support organization willing or able to take on the job of certifying each SSH daemon.

Our approach permits a fully decentralized SSH PKI that supports such users, by permitting each one to be their own CA.

When the user is at his home site and has trusted access to the server public keys, he runs a configurator program that collects from the user a passphrase, the name(s) of the desired target servers, and a path to a trusted store of their public keys. The configurator then produces a *hashstore* file containing a list of server names, and (for each server) the keyed MAC, generated under a symmetric key derived from the passphrase, of the

tuple of server name and public keys. The user then places this file in some publicly accessible place, such as in their home directory on the Web.

When the user then wishes to use SSH from a remote client, he initiates the connection to his home machine (e.g., `ab.cs.foo.edu`). Then, `ab.cs.foo.edu` sends its public key to the client machine. The modified SSH client prompts the user for the URL of the hashstore. The modified client fetches the hashstore and extracts the keyed MAC for `ab.cs.foo.edu`. The modified SSH client prompts the user for his passphrase. The client then derives a symmetric key from this passphrase, and uses it to generate the keyed MAC for the alleged public key. The generated MAC is compared with the MAC received from the web page. If the two values match, the client proceeds with SSH.

Variations are possible. For example, we could take the decentralization even further and, rather than having the user remember the current global name for each target host, the modified client could use the hashstore to obtain an IP address or other external name for the personal hostname the user typed (thus avoiding the `foo.bar.com` problem our colleague encountered).

Semi-centralized In some scenarios, it might be reasonable for an enterprise to set up a CA that reliably signs certificates for SSH daemons at servers.

Our approach also permits such semi-centralized approaches. The SSH protocol will already have the server send this certificate to the client. What we need to do is provide the client with a certification path to verify this information; a minimal one might be a certificate for the user's home enterprise's root CA.

In this case, the enterprise admin set up an LDAP database, and runs a configurator tool that populates the database with a record, for each user, containing a this certification path and a keyed MAC for it, generated via the user's passphrase.

When the user then wishes to use SSH from a remote client, he initiates the connection to his home machine (e.g., `ab.cs.foo.edu`). Then, `ab.cs.foo.edu` sends its certificate to the client machine. The client prompts the user for the location of the LDAP. The client contacts the LDAP, and sends the user's name. The LDAP server looks up the username and sends back the certification path and the hashed CA certificate corresponding to that. The modified SSH client prompts the user for his passphrase. The client then derives a symmetric key from this passphrase, and uses that key to generate a keyed MAC of the certification path received. If the MACs match, the client then validates the server certificate using this certification path. The client then proceeds with SSH.

Again, many variations are possible here. For example, the CA need not coincide with the party setting up the LDAP; alternatively, an individual user could set up his Web-based hashstore (as in the decentralized approach above) to include the public keys of CAs he chooses to trust.

In this semi-centralized approach, the only role of the CA is to certify SSH daemons for its user population—which is probably not a high-volume task. Since the CA does not certify users, we anticipate that revoking and re-issuing certificates will be infrequent. Avoiding the need for a CA certificate issued by a standard trust root also reduces expense and hassle. Thus, we eliminate many potential performance and exposure issues—this machine will just not be used that often.

Furthermore, by easily allowing a user to go right to a root, this approach permits a trust hierarchy that consists of a forest of small trees. The user specifies the tree root. This eliminates the hassle of intermediate certificates (cross-certificates or bridge certificates) that would be generated if we wanted interoperability between multiple Certificate Authorities within one realm. This also provides increased resilience; compromising the security of one CA will not compromise the whole PKI.

Certificates and Expiration Since our solution's "small-delta" constraint dictated simplicity, we opted for long lifespans on server certificates and hope for minimal certificate revocation. We considered delegated path discovery and validation [6], which would enable the client to offload checking CRLs and other such duties to a remote server, but decided against it for simplicity.

In the worst case, supporting revocation of the trust root (and preventing replay attacks) would require a way to "revoke" old keyed hashes. One approach would be to have users change passphrases; another would be to have the LDAP know the users' passphrases (a risk).

We opted for X509v3 certificates. The reasons for that are multifold. First of all, the X.509 standard [1] constitutes a widely accepted basis for such an infrastructure. Secondly, Microsoft Certificate Store and OpenSSL libraries are both interoperable with x509v3 certificates. Thirdly, x509v3 certificates support extensions that can be added into a certificate, which can then uniquely identify a certificate on the basis of its IP address extension.

4.3 Security

The security of the above approaches stems from the basic fact that the user can use a passphrase as a shared secret key to create a hash.

Semi-centralized The semi-centralized approach starts by having the server `ab.cs.foo.edu` send the server certificate to the client machine. An attacker who has a different server certificate issued by the same certificate authority can intercept the server certificate. Suppose she wants to pose as `ab.cs.foo.edu`. She replaces the server certificate with her own certificate. Certificate verification would fail in such a scenario because when the SSH client looks at the uniquely identifying server name of the certificate, it would not be the one it expected. The server name would not match. If the attacker forges it to be the same, she would not be able to modify her signature to match the forged extension, as the CA generated the signature, therefore the certificate verification would fail again. The client would verify the signature of the attacker using the CA's public key and match it with the credentials provided which would not be the same. This establishes the fact that the attacker cannot forge a suitable certificate, even if she has a certificate issued by the same certificate authority.

The user types in the URL/LDAP address, which is connected to an LDAP server; the client sends the user name to the LDAP. A malicious user can intercept the username and send back a different hash of a modified certification path, and a modified certification path itself. However when the client uses the user passphrase to generate

the hash for the modified certification path, it would not match the hash received. The attacker would not know the password used by the user. Essentially, the only way to crack this protocol is by cracking the password used. That can be done by dictionary attacks or by other techniques, such as social engineering. Therefore it is important that the user selects a long passphrase and fulfills the requirements of a good passphrase to ensure that this protocol is secure.

Revocation Suppose the server certificate changes. In such a scenario, an attacker could use the old invalid server certificate from a previous session and successfully pose as the server. The user would receive the old certificate, and he would retrieve the hashed CA certification path, and the CA certification path from the LDAP server. As the CA's public key has not been modified, the user would validate the invalid server certificate, and send his username and plaintext password to the attacker.

There are two ways to avoid this problem. Firstly, the usage of Certificate Revocation Lists or OCSP (e.g. [4]) can inform clients that a certificate has been revoked. In that case, the client would check whether the certificate has been revoked or not. This approach requires addition of components to the existing protocol, which would query the status of the certificate, retrieved at the client end and manage or store certificate revocation [1] information at the server end. A simpler solution to the problem is that server certificates issued are irrevocable. What that means is that once a server is assigned a certificate, it cannot be changed until it expires. Once the server certificate is expired it would be recertified. This option does not complicate the protocol. This protocol is secure as long as the private key corresponding to the certificate is protected and safe.

4.4 Decentralized Approach

The decentralized approach starts out by having the server `ab.cs.foo.edu` send its public key to the client machine. Suppose an attacker intercepts and replaces it with her own public key. Once the client receives the public key, it retrieves the keyed-hash of the public key. It hashes the public key and compares it with the hash received. The two hashes would not match.

Suppose the attacker now attempts to send her own keyed hash instead of the one stored at the place the user specified. To be able to generate a valid hash, the attacker needs to know the user's passphrase. Therefore if she replaces a valid hash with one of her own, that hash would not match to the one that the client would produce using the user's passphrase on the spoofed public key received.

Attacks due to DNS Spoofing are also defied by the verification of the host key. Suppose that the user logs in on a client machine for the first time and types in `ab.cs.foo.edu` that maps to `129.172.111.4`. However, an attacker spoofs it so that the user attempts to connect to `129.172.111.5` instead of `129.172.111.4`. The fact that the attacker can only possess the public key of the server—and not the private key—implies that she cannot generate the signatures that validate the payload during the key exchange, therefore he can not successfully establish a shared secret key which follows a successful server authentication at the transport layer.

Revocation Suppose the server public key changes. In such a scenario, the client would be vulnerable to a replay attack. In a simpler model, the user can be informed via email, as soon as possible, of the change in the server key so that he can update his web page repository of hashes. This methodology introduces a window of risk; however, in the interest of simplicity (and in the hope that server certificates do not change often), we believe that it should suffice.

4.5 Drawbacks

Our solution does have some disadvantages.

First, our use of a hash keyed from a user-memorized passphrase introduces a risk of offline dictionary attack on the passphrase. For now, our defense here is to encourage users to use sufficiently long passphrases to minimize this risk. (Potentially, we could also augment the hash-fetch protocol to include temporary symmetric keys known only by the client and the SSH server keyholder—requiring an attacker to actively impersonate a server in order to gather material for a dictionary attack.)

Secondly, our solutions may require user participation in dealing with revocation of SSH server key pairs. Here, we believe that the relative infrequency of this activity—combined with the fact that the default scenario requires the users to actively participate in SSH key pairs anyway—will reduce the significance of this requirement.

Both of these issues remain areas for future work.

Of course, our principal disadvantage that we require that borrowed client machines have an alternate SSH client.

5 Implementation

5.1 Overview

We prototyped the fully decentralized solution above. We also have hooks in place for the semi-centralized solution (and some other techniques we discuss in Sect. 6).

The design and code that was written to develop the prototype is open source and modifies the SSH protocol.

At the time of our experiments, the existing SSH protocol supported by OpenSSH had the bare skeletal structure to support certificates that does not directly support certificate verification. OpenSSL(0.9.6c) has created routines and callback functions that can be used with much ease to verify certificates. We added a `FirstTime` authentication method to carry out our decentralized approach to verifying keys. Figure 1 sketches an architectural block diagram that explains how we extended the SSH client.

For a keyed MAC, we chose HMAC [3], which is built on a cryptographic hash function, typically MD5 or SHA-1 (we chose SHA-1). We derive the symmetric key from a 20-character passphrase entered by the client.

5.2 Choice of Codebase

At the time of our experiments, open-source OpenSSH clients existed for Linux, and the only fully-deployed open source SSH server was provided by OpenSSH and was

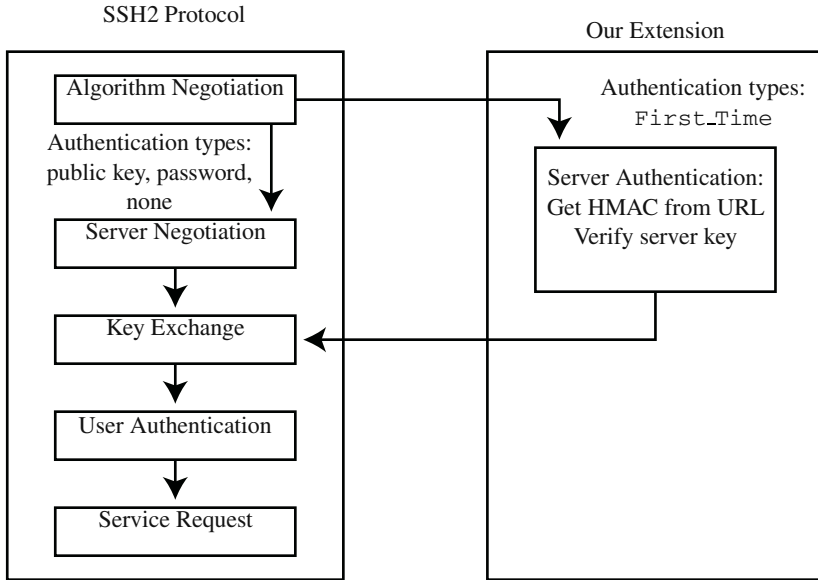


Fig. 1. Architectural block diagram of the `First_Time` method.

distributed freely with Red Hat Linux. NetworkSimplicity.com provides a Windows port for the OpenSSH client and server. However, using this codebase would not give us a graphical user interface.

TeraTerm SSH clients were interoperable with OpenSSH servers and provided a Graphical User Interface that makes it easy to use on Windows platform. The TeraTerm SSH client is an extension to TeraTerm Telnet client, which is also open source.

For our work, we chose the TeraTerm SSH client, primarily because we felt that a graphical Windows client would be more useful for traveling users.

5.3 Application Interface

Figure 2 shows the main entry. To use the fully decentralized option, the user selects the “Enable Secure First Time Authentication” option and clicks “Configure.” Once the user enters the hostname he is prompted with the User Interface displayed at left in Figure 3, which prompts the user for the hashstore.

The client receives the public key/server certificate and is prompted to enter the hash password (Figure 3, right). If the password authenticates the hash, SSH authentication proceeds as normal. (Figure 4).

5.4 Configurator

Our configurator utility is an independent program that sets up the hashstore. The hashstore contains a list of the machines, that the user accesses remotely, and their corresponding Hash values.

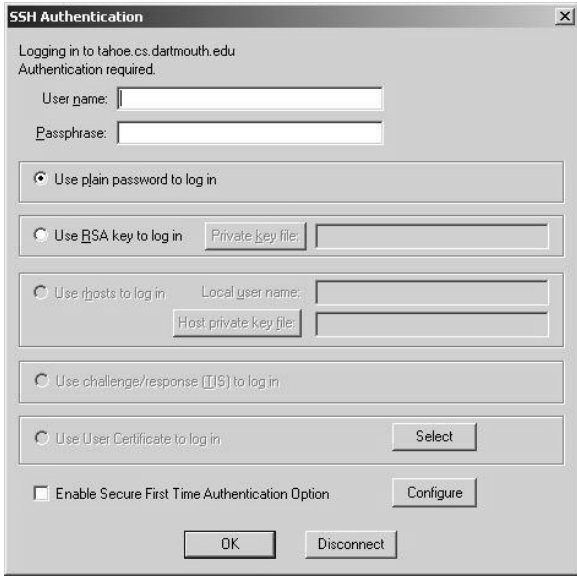


Fig. 2. The main screen



Fig. 3. At left, the client requests URL entry for hashstore; at right, the client requests the passphrase to generate the HMAC.

This program takes the server public key used for SSH sessions, concatenates it with the server name and then generates the hash and stores them in a file called hashstore. The file is then placed on a hosted Web site. Figure 5 shows sample of the output when this program runs.

5.5 Code Availability

All the code that was used and modified in this project is open source and can be downloaded from <http://www.cs.dartmouth.edu/~sws/yasir.thesis>. The binaries for the modified TeraTerm SSH clients are also available. The code contains all the needed OpenSSL libraries, crypt32 libraries from Microsoft platform SDK needed for certificate management. A readme.txt file explains how to setup the client, and the hash generation tool.



Fig. 4. If the retrieved HMAC matches the public key the server sent, then the server has been authenticated.

```

$./linuxConf
This program generates the HMAC Hashes for the public
keys of host machines. This program is to be run on the local
machine whose public key is to be hashed

Enter the host name of the machine: foo.bar.com
----String to be Hashed---
foo.bar.com 1024 35
151052055305049513313873325220639580170308824866279975903865
17065659085280834883625163014772300422177712761876894150979694040698019214365467
07381986625813796034665095626099575794106722934274328194857789445170395185623181
06109502073419576490304279566550418746874044392631514426610329187514765207346921
326949181
Enter the pass phrase to hash:
MySecretPassPhrase%l

---Hash stored in hashstore.txt---
foo.bar.com 1024 35 MD:_Y%S (non ascii characters not displayed)
$

```

Fig. 5. Sample output of the configurator, which generates HMAC values of the public keys, to server as the user’s trust roots from remote locations. (The use of the keyed MAC ensures integrity without a universal PKI.)

6 Conclusions and Future Work

Our solution brings public-key security to SSH in a a flexible and scalable way—and (when one considers the decentralized approach) constitutes a datapoint for a useful “PKI” with neither CA nor certificate.

The ease of use and deployment of the “decentralized non-CA approach” distinguishes our solution from one based on a traditional hierarchical PKI.

Currently, both academia and industry strive to develop standardized protocols that would make the deployment of PKI relatively manageable; however, a look at the deployed technology base shows that we’re not quite there yet. Most small enterprises do not need to invest and develop a hierarchical trust model for a PKI. Our decentralized approach is ideal for such small-scale corporate environments. The users do not have to learn the how to use user certificates and the system administrators do not have to set up a PKI. In contrast, our centralized CA approach is suited for larger networks with several users. It develops a PKI with a minimal set of components.

In future work, we would like to take the prototype to another level of completion. Right now, the user interface is not completely clear; and arguably, a secure client for

traveling users should be aggressive about *not* retaining validated server keys. We would also like to extend the code to handle more pieces of the solution space; for example, the current code does not support the semi-centralized approach (beyond connecting to an LDAP server). We also would like to strengthen the way that the HMAC keys are derived from passphrases. Developing a Linux version would also be useful. Exploring *visual hashing* [5] as an alternative way for a human user to authenticate complex data would also be interesting.

References

1. Carlisle Adams and Stephen Farrell. "Internet X.509 Public Key Infrastructure Certificate Management Protocols." IETF RFC 2510, March 1999.
2. Daniel J. Barrett and Richard E. Silverman. *SSH: The Secure Shell, The Definitive Guide*. O'Reilly & Associates. 2001.
3. H. Krawczyk, M. Bellare, R. Canetti. "HMAC: Keyed Hashing for Message Authentication." RFC 2104, February 1997.
4. Michael Myers, Rich Ankney, Carlisle Adams, Stephen Farrell, and Carlin Covey. "Online Certificate Status Protocol, version 2." Internet Draft, March 2001.
5. Adrian Perrig and Dawn Sogn. "Hash Visualization: A New Technique to Improve Real-World Security." *International Workshop on Cryptographic Techniques and E-Commerce*. 1999
6. Denis Pinkas, Russ Housley. "Delegated Path Validation and Delegated Path Discovery Protocol Requirements." Internet Draft, February, 2002.
7. J. Schlyter, W. Griffin. "Using DNS to Securely Publish SSH Key Fingerprints." Secure Shell Working Group, Internet Draft. September 2003.
8. Dawng Song, David Wagner, Xuqing Tian. "Timing Analysis of Keystrokes and Timing Attacks on SSH." *10th USENIX Security Symposium*. 2001.
9. Siva Sai Yerubandi, Weetit Wanalertlak. "SSH1 Man in the Middle Attack." Oregon State University. <http://islab.oregonstate.edu/koc/ece478/project/2002RP/YW.pdf>. 2002.
10. T. Ylonen, D. Moffat. "SSH Protocol Architecture." Network Working group, Internet Draft, October 2003.
11. T. Ylonen, D. Moffat. "SSH Connection Protocol." Network Working group, Internet Draft, October 2003.
12. T. Ylonen, D. Moffat. "SSH Transport Layer Protocol." Network Working group, Internet Draft, October 2003.
13. T. Ylonen, D. Moffat. "SSH Authentication Protocol." Network Working group, Internet Draft, September 2002.

What Is Possible with Identity Based Cryptography for PKIs and What Still Must Be Improved

Benoît Libert^{1,2*} and Jean-Jacques Quisquater¹

¹ UCL Crypto Group

Place du Levant, 3. B-1348 Louvain-La-Neuve. Belgium

{libert,jjq}@dice.ucl.ac.be

² Laboratoire d'Informatique de l'École Polytechnique (LIX)

F-91128 Palaiseau CEDEX, France

Abstract. This paper is a survey of the advantages that the use of identity based cryptosystems can provide to PKIs. Since its introduction by Shamir in 1984, a couple of breakthroughs have been achieved in this area: namely, several identity based encryption (IBE) and identity based signature (IBS) schemes were proposed as well as hierarchical extensions of these while various special purpose identity based signatures also appeared. Interestingly, this kind of public key encryption and digital signature schemes can easily be turned into other kinds of primitives such as key evolving cryptosystems. This paper summarizes all the interesting properties of ID-based cryptography and recalls the aspects for which it still has to be improved.

Keywords. identity based cryptography, pairings, PKIs.

1 Introduction

Since the appearance of public key cryptography in 1976, the threat of "man-in-the-middle attacks" has been a great concern for the research community that was led to devise certification methods to provide users with confidence in the authenticity of the public keys they are using. These guarantees took the form of digital certificates signed by trusted entities called Certification Authorities (CAs) which aimed at vouching for the fact that a given public key actually belongs to its alleged owner. It was the birth of Public Key Infrastructures (PKIs) that deployed mechanisms to manage these digital certificates throughout the lifetime of their corresponding keys. Unfortunately, these certificate-based infrastructures turned out to be very heavy to deploy, cumbersome to use and non-transparent for the user. Indeed, trust problems arise when a digital certificate is signed by an authority whose public key is not already trusted by the user of the certified public key: in such a case, the user is led to validate an

* This author is supported by the DGTRE's First Europe Project.

entire chain of digital certificates before acquiring confidence in the authenticity of a given key and, furthermore, finding such a chain of certificates between the enquired key and a trusted one is not a trivial problem. The treatment of certification paths has also been a critical issue in PKIs and a lot of softwares like web browsers are currently still unable to deal with it: as an example, at the establishment of an SSL connection, the server's certificate has to be signed by an authority that is already trusted by the client's browser.

Other problems with PKIs is the fact that certificates are not be delivered for free in many situations and their cost is making owner of public keys reluctant to enquire for them and, from a robustness point of view, their lack of fault tolerance: when a private key exposure happens, no easy solution allows to repair it nor to limit the damage it involves: the corresponding certificate must be invalidated by using black lists called Certificate Revocation Lists (CRLs) that must be periodically checked by users who want to ensure that the key they are about to use has not been compromised.

In order to bypass the trust problems encountered in conventional Public Key Infrastructures, Shamir introduced in 1984 ([43]) the concept of identity based cryptography where a public key can be a binary string identifying its owner non-ambiguously (e.g. an e-mail address, an IP address combined to a user-name, a social security number,...). The motivation of this kind of scheme was to simplify key management and remove the need of public key certificates as much as possible: since a key is the identity of its owner, there is no need to bind them by a digital certificates while a public repository containing a list of user names and their associated public key becomes useless since public keys are human-memorizable. End users do not have to enquire for a certificate for their public key. The only things that still must be certified are the public keys of trusted authorities called private key generators (PKG) that have to generate private keys associated to users' identities thanks to their secret key (unlike conventional public key schemes, users do not generate their key pair themselves). This does not completely remove the need of certificates but, since many users depend on the same authority, this need is drastically reduced.

One inconvenience of these systems is their inherent key escrow feature. Indeed, since trusted authorities called private key generators (PKG) have to deliver private keys to users after having computed them from their identity information and from a master secret key, these so-called PKGs are able to sign messages on behalf of or to decrypt ciphertexts intended to any user depending on them. Several practical solutions for identity based signatures (IBS) have been devised since 1984 ([20],[26], [42],etc.) but finding a practical identity based encryption scheme (IBE) remained an open challenge until 2001 when Boneh and Franklin ([7]) proposed to use bilinear maps (the Weil or Tate pairing) over elliptic curves to achieve an elegant identity based encryption method. Other identity based signature and key agreement schemes based on pairings were proposed after 2001 ([13],[44],[27],...). We also mention the existence of many other proposals for ID-based cryptographic protocols from pairings. They are not discussed here but are referenced in [2].

In this survey we give an overview of the solutions offered by Identity Based Cryptography technologies to provide PKIs with a greater flexibility as well as a better fault-tolerance. We first recall the Boneh-Franklin IBE and Boyen's signature/encryption scheme ([11]) before describing the hierarchical extension of IBE proposed by Gentry and Silverberg ([24]) and showing how it provides fault-tolerance. We then discuss about some solutions to remove the inherent key escrow (that might be especially undesirable for signatures) from the ID-based methodology and we end the paper by a summary of the advantages of the Identity Based paradigm when compared to its variants such as Girault's self-certified model ([25]), Gentry's certificate-based solution ([23]) or the Al-Ryami-Paterson certificateless proposal ([3]). We also outline the currently unsolved problems with identity based cryptosystems that could still hamper their wide use in security infrastructures.

2 Preliminaries

2.1 Components of Identity Based Cryptosystems

Basically, an identity based cryptosystem is made of four algorithms that are the following ones.

Setup: is an algorithm run by a PKG that takes as input a security parameter to output a public/private key pair (P_{pub}, mk) for the PKG (P_{pub} is its public key and mk is its master key that is kept secret).

Keygen: is a key generation algorithm run by a PKG: it takes as input the PKG's master key mk and a user's identity ID to return the user's private key d_{ID} .

Encrypt: in the case of identity based encryption, the third algorithm is an encryption algorithm that is run by anyone and takes as input a plaintext M , the recipient's identity and the PKG's public key P_{pub} to output a ciphertext C .

Decrypt: in the case of ID-based encryption, the last algorithm is then a decryption algorithm that takes as input the ciphertext C and the private decryption key d_{ID} to return a plaintext M .

In the case of identity based signatures, the last two algorithms are

Sign: given a message M , the PKG's public key and a private key d_{ID} , the signature generation algorithm generates a signature on M .

Verify: is a signature verification algorithm that, given an alleged signature σ on a message M for an identity ID , outputs either 1 or 0 depending on whether the signature is acceptable or not.

2.2 Overview of Pairings

Pairings are bilinear maps that can be defined over cyclic elliptic curve subgroups for a particular kind of curves. These curves were first thought to be unsuitable

for cryptographic purposes because of the MOV reduction ([35]) from the elliptic curve discrete logarithm problem (ECDLP) to the discrete logarithm in a finite field. In 2000, Joux ([29]) showed how to use pairings over these curves in constructive cryptographic applications. Namely, he showed how to devise a one-round tripartite Diffie-Hellman protocol using them. Since that seminal paper, pairings provided a couple of other applications (see [2]). One of the most important of them was Boneh and Franklin's identity based encryption protocol ([7]) that is recalled in the upcoming sections.

Let us consider groups \mathbb{G}_1 and \mathbb{G}_2 of the same prime order q . We need a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfying the following properties:

1. Bilinearity: $\forall P, Q \in \mathbb{G}_1, \forall a, b \in \mathbb{Z}_q^*$, we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
2. Non-degeneracy: for any $P \in \mathbb{G}_1$, $\hat{e}(P, Q) = 1$ for all $Q \in \mathbb{G}_1$ iff $P = \mathcal{O}$.
3. Computability: for all $P, Q \in \mathbb{G}_1$, $\hat{e}(P, Q)$ is efficiently computable.

It is shown in ([7]) how to derive such admissible bilinear maps from the Weil and the Tate pairings over supersingular elliptic curves.

2.3 The Boneh-Franklin Identity Based Encryption Scheme

We describe the basic version of the scheme. This version is only provably secure against chosen-plaintext attacks and has some similarities with El Gamal's cryptosystem. Boneh and Franklin showed that applying the Fujisaki-Okamoto generic transformation ([22]) allows turning this basic scheme into a chosen-ciphertext secure one in an extended security model (see [7] for details). Their scheme, proposed in 2001 (roughly at the same time as the one described by Cocks in [15]) turns out to be the first one to be really secure and practical.

Setup: given a security parameter k , the PKG chooses groups \mathbb{G}_1 and \mathbb{G}_2 of prime order $q > 2^k$, a generator P of \mathbb{G}_1 , a master key $s \in \mathbb{Z}_q^*$ and the associated public key $P_{pub} = sP$. The plaintext space is $\mathcal{M} = \{0, 1\}^n$ for a fixed n while $\mathcal{C} = \mathbb{G}_1^* \times \{0, 1\}^n$ is the ciphertext space. Hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n$ are also chosen. The system's public parameters are

$$\text{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2).$$

Keygen: given identity $ID \in \{0, 1\}^*$, the PKG computes $Q_{ID} = H_1(ID) \in \mathbb{G}_1$ and $d_{ID} = sQ_{ID} \in \mathbb{G}_1$ that is given to the user as a private key.

Encrypt: to encrypt a message $M \in \mathcal{M}$, the sender uses the recipient's public identifier $Q_{ID} = H_1(ID) \in \mathbb{G}_1$ as follows:

1. He/she picks a random $r \in \mathbb{Z}_q^*$.
2. He/she computes $g_{ID} = \hat{e}(Q_{ID}, P_{pub}) \in \mathbb{G}_2$.

The ciphertext $C = \langle rP, M \oplus H_2(g_{ID}^r) \rangle$ is sent to the receiver.

Decrypt : given a ciphertext $C = \langle U, V \rangle$, the receiver uses his/her private key d_{ID} to decrypt by computing $M = V \oplus H_2(\hat{e}(d_{ID}, U))$.

The protocol's consistency is easy to check. Indeed, if the sender correctly encrypted the message, we have $U = rP$ and

$$\hat{e}(d_{ID}, U) = \hat{e}(sQ_{ID}, rP) = e(Q_{ID}, P_{pub})^r = g_{ID}^r.$$

It is shown in [7] that, when padded with the Fujisaki-Okamoto transform ([22]), the above scheme is secure against adaptive chosen-ciphertext attacks.

In the version described above, the crucial information is the PKG's master key: all the system's privacy is compromised if that master key is ever stolen by an attacker. In order to avoid having a single point of failure and remove the built-in key escrow, Boneh and Franklin showed it was possible to split the PKG into several partial PKGs in such a way that these partial PKGs jointly generate a discrete logarithm key pair in a threshold fashion and each of them eventually holds a share of the master key. Users then have to visit a minimum of t -out-of- n honest PKGs to obtain a share of their private decryption key. These shares can then be recombined into a full decryption key thanks to a Lagrange interpolation as in Shamir's secret sharing scheme. An alternative to this approach was suggested by Chen et al. ([14]) who imagined a setting with n different PKGs, each having their own master key/public key pair and issuing private keys associated to users' identities independently: a user's full private key was simply the sum of the n received independent private keys while the scheme full public key was the sum of the n PKGs' public keys. These latter two approaches have the drawback to give the same role to the different PKGs that have to independently check the user's identity before delivering him/her a partial private key. This might be a burden in some situations and it hampers any centralized key issuing policy among the PKGs. In section 4, we will discuss about some alternatives to these key escrow prevention methods.

2.4 Identity Based Signature/Encryption

At Crypto 2003 ([11]), Boyen described a practical two-layer signature/encryption design to achieve identity based authenticated encryption. His scheme is provably chosen-ciphertext secure, provides detachable and non-repudiable signatures that are unlinkable to the corresponding original ciphertext while the recipient is convinced that the message was signed and encrypted by the same user. The scheme is suitable for applications requiring both confidentiality and authentication and, furthermore, ciphertexts are fully anonymous (i.e. they provably do not leak any information about their intended recipient nor about who created them). It also supports encryption for multiple recipient with a single signature. This result was obtained by a secure randomness re-use in the Cha-Cheon ([13]) probabilistic ID-based signature followed by a deterministic encryption. The scheme consists of the 6 algorithms depicted on figure 1.

From an efficiency point of view, the protocol is comparable to other monolithic signcryption schemes ([34],[32],[37]) but, unlike the latter schemes, it provides detachable signatures that cannot be linked to ciphertexts: while a sender cannot repudiate a detached genuine signature, he can always deny a given ciphertext: the receiver is the only entity who can ensure that a signed message

was encrypted by the user who signed it. When compared to any naive signature/encryption composition, Boyen's scheme produces shorter final ciphertexts.

Setup: is the same as in Boneh and Franklin's IBE except that five hash functions are needed: $H_0 : \{0, 1\}^* \rightarrow \mathbb{G}_1$, $H_1 : \{0, 1\}^{n_0} \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^\ell$, $H_3 : \mathbb{G}_2 \rightarrow \mathbb{Z}_q^*$ and $H_4 : \mathbb{G}_1 \rightarrow \{0, 1\}^{n_0+n_1}$ where ℓ is the size of the representation of \mathbb{G}_1 's elements, n_0 is the size of plaintexts and n_1 the maximum length of identifiers.

Keygen: is the same as in IBE.

Sign: given a message M and the sender's private key d_{ID_A} ,

1. pick $r \in \mathbb{Z}_q^*$, compute $U = rQ_{ID_A} \in \mathbb{G}_1$ and $h = H_1(M, U) \in \mathbb{Z}_q^*$,
2. set $V = (r + h)d_{ID_A} \in \mathbb{G}_1$,
3. return $\langle M, r, U, V, ID_A, d_{ID_A} \rangle$.
to Encrypt

Encrypt:

given $\langle M, r, U, V, ID_A, d_{ID_A} \rangle$
and the recipient's identity

$Q_{ID_B} = H_0(ID_B) \in \mathbb{G}_1$,

1. compute $\omega = \hat{e}(d_{ID_A}, Q_{ID_B})$
and $\lambda = H_3(\omega) \in \mathbb{Z}_q^*$,
2. set $X = \lambda U \in \mathbb{G}_1$ and
 $Y = V \oplus H_2(\omega^{\lambda r}) \in \{0, 1\}^\ell$,
3. compute $Z = (M || ID_A) \oplus H_4(V)$
 $\in \{0, 1\}^{n_0+n_1}$ and return
the final ciphertext $\langle X, Y, Z \rangle$.

Decrypt: given a ciphertext

$\langle X, Y, Z \rangle$ and his private key d_{ID_B} ,

1. compute $V = Y \oplus H_2(\hat{e}(X, d_{ID_B}))$
and then $(M || ID_A) = Z \oplus H_4(V)$,
2. compute $Q_{ID_A} = H_0(ID_A) \in \mathbb{G}_1$,
 $\omega = \hat{e}(Q_{ID_A}, d_{ID_B}) \in \mathbb{G}_2$ and then
 $\lambda = H_3(\omega) \in \mathbb{Z}_q^*$.
3. compute $U = \lambda^{-1}X \in \mathbb{G}_1$ and
return $\langle M, ID_A, U, V \rangle$ to Verify

Verify: given $\langle M, ID_A, U, V \rangle$,

Return 1 if

$\hat{e}(P, V) = \hat{e}(P_{pub}, U + hQ_{ID_A})$.
where $h = H_1(M, U) \in \mathbb{Z}_q^*$
and $Q_{ID_A} = H_0(ID_A) \in \mathbb{G}_1$
and 0 otherwise.

Fig. 1. Identity Based Signature/Encryption

2.5 Hierarchical Identity Based Cryptography

A shortcoming of the Boneh-Franklin identity based encryption scheme is that in a large network, the PKG's key generation task rapidly becomes a bottleneck when many private keys have to be computed and secure channels have to be established to transmit them to their legitimate owner. To overcome this problem, a solution is to setup a hierarchy of PKGs in which each PKG only computes private keys for entities (other PKGs or end-users) immediately below it in the

hierarchy. In such hierarchies, entities are represented by a vector of identifiers (i.e. a concatenation of their identifier to those of all their ancestors' ones: for example a child of $\langle ID_1, \dots, ID_i \rangle$ has an address $\langle ID_1, \dots, ID_i, ID_{i+1} \rangle$) instead of a single identifier as in the Boneh-Franklin scheme.

The Gentry-Silverberg Hierarchical Scheme

In this section, we give an example, proposed by Gentry and Silverberg ([24]), of such an hierarchical scheme that can be viewed as a scalable extension of Boneh and Franklin's proposal (both schemes are identical if the hierarchy has a single level). Unlike another hierarchical scheme proposed by Horwitz and Lynn ([28]), this one supports an arbitrary number of levels ℓ . Lower-level PKGs (i.e. PKGs other than the Root PKG located at the top of the hierarchy) generate private keys for their children by using some information coming from their ancestors together with a private information that is only known to them. Each of them then adds some information to the secret parameters of their children.

In our notation, $Level_i$ is the set of entities at level i , $Level_0$ denotes the sole Root PKG. The simplified version of the scheme is made of the following algorithms.

Root Setup: given a security parameter k , the root PKG

1. generates groups \mathbb{G}_1 and \mathbb{G}_2 of prime order q and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ and picks a generator $P_0 \in \mathbb{G}_1$,
2. chooses $s_0 \in \mathbb{Z}_q$ and sets $Q_0 = s_0 P_0$,
3. chooses hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n$ for some n denoting the size of plaintexts.

The ciphertext space is $\mathcal{C} = \mathbb{G}_1^t \times \{0, 1\}^n$ where t is the level of the ciphertext's recipient in the hierarchy. The system-wide parameters include **params** := $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2)$. The master key is $s_0 \in \mathbb{Z}_q$.

Lower Level Setup: An entity E_t at level $Level_t$ randomly picks $s_t \in \mathbb{Z}_q$ and keeps it secret.

Keygen $_{t-1}$: at level $Level_t$, we consider an entity E_t of address (ID_1, \dots, ID_t) , where (ID_1, \dots, ID_i) , for $1 \leq i \leq t-1$, is the address of its ancestor at $Level_i$. Let S_0 be the unit element of \mathbb{G}_1 . At $Level_{t-1}$, the father E_{t-1} of E_t generates E_t 's private key as follows:

1. It computes $P_t = H_1(ID_1, \dots, ID_t) \in \mathbb{G}_1$.
2. It computes E_t secret point $S_t = S_{t-1} + s_{t-1}P_t = \sum_{i=1}^t s_{i-1}P_i$ and transmits it to E_t . S_t then depends on secret elements $S_{t-1} \in \mathbb{G}_1$ and $s_{t-1} \in \mathbb{Z}_q$ of E_{t-1} .
3. E_{t-1} also transmits $Q_i = s_i P_0 \in \mathbb{G}_1$ to E_t for $1 \leq i \leq t-1$ (it must have computed $Q_{t-1} = s_{t-1} P_0$ itself from its secret s_{t-1} and have received Q_0, \dots, Q_{t-2} from its ancestors).

E_t 's private key is $(S_t, Q_1, \dots, Q_t, s_t)$. The part $(S_t, Q_1, \dots, Q_{t-1})$ is received from its father E_{t-1} and it generates the components s_t and $Q_t = s_t P_0$ itself.

Encrypt: to encrypt a message $M \in \mathcal{M}$ for an entity E_t of address (ID_1, \dots, ID_t) ,

1. Alice computes $P_i = H_1(ID_1, \dots, ID_i) \in \mathbb{G}_1$ for $1 \leq i \leq t$.
2. She randomly picks $r \in \mathbb{Z}_q$.
3. The ciphertext is $C = [rP_0, rP_2, \dots, rP_t, M \oplus H_2(g^r)]$ with $g = \hat{e}(Q_0, P_1)$.

Decrypt: E_t receives $C = [U_0, U_2, \dots, U_t, V] \in \mathcal{C}$. To decrypt it, he/she computes

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_t)}{\prod_{i=2}^t \hat{e}(Q_{i-1}, U_i)}\right) = M.$$

The bilinearity of the map \hat{e} allows verifying the consistency of the scheme:

$$\begin{aligned} \hat{e}(U_0, S_t) &= \hat{e}(P_0, P_1)^{rs_0} \hat{e}(P_0, P_2)^{rs_1} \dots \hat{e}(P_0, P_t)^{rs_{t-1}} \\ &= \hat{e}(Q_0, P_1)^r \hat{e}(Q_1, P_2)^r \dots \hat{e}(Q_{t-1}, P_t)^r \\ &= g^r \hat{e}(Q_1, U_2) \dots \hat{e}(Q_{t-1}, U_t) \end{aligned}$$

and $\frac{\hat{e}(U_0, S_t)}{\prod_{i=2}^t \hat{e}(Q_{i-1}, U_i)} = g^r$ for the g computed by Alice at the encryption.

The above version of the scheme is a simplified one satisfying only the security notion of one-wayness against chosen-plaintext attacks. To convert it into a chosen-ciphertext secure one, the Fujisaki-Okamoto generic transformation ([22]) is simply applied to it. Unlike the 2-level solution proposed by Horwitz and Lynn in 2002, the resulting scheme provably resists to a collusion between any number of dishonest users (i.e. a set of users combining their private information in an attempt to threaten the confidentiality of messages sent to a honest user). It is showed in [24] how to turn the above encryption scheme into a hierarchical identity based signature and how to shorten the ciphertexts produced by the scheme (as well as the signatures outputted by the derived hierarchical signature issuing protocol).

As mentioned in [24], it is interesting to notice that key escrow is somewhat restricted here: only a user's father knows his/her complete private key (because of lower-level secret information) and the other PKGs are unable to decrypt ciphertexts intended to him/her.

Exposure-Resilience for Free

In [19], Dodis and Yung showed that the scheme of Gentry and Silverberg naturally and very simply provides a method to achieve exposure-resilience. They noticed that, among the t pieces of secret information held by an entity at *Level* t , any subset of $t - 1$ of them does not reveal any information to an adversary. This reduces the requirement for secure storage and, furthermore, exposure-resilience can be achieved by letting each user perform periodic refreshes of its secret key. These refreshes are transparent to the outside world since the refreshed private

keys are exactly as functional as the previous ones. The free exposure-resilience is obtained as follows: recall that the private key of a user at $Level_t$ is given by

$$S_t = \sum_{i=1}^t s_{i-1}P_i, \quad Q_1 = s_1P_0, \dots, Q_{t-1} = s_{t-1}P_0.$$

Among the values s_0, \dots, s_{t-1} , only s_0 is fixed by the public key $Q_0 = s_0P_0$: s_1, \dots, s_{t-1} can be arbitrary. A private key can thus be refreshed by randomly picking $s'_1, \dots, s'_{t-1} \in \mathbb{Z}_q$ to update S_t and Q_1, \dots, Q_{t-1} as

$$S_t := S_t + \sum_{i=2}^t s'_{i-1}P_i, \quad Q_i := Q_i + s'_iP_0 \quad \text{for } 1 \leq i < t$$

Any subset of $t-1$ out of t secret "old" values S_t, Q_1, \dots, Q_{t-1} does not reveal any information about the freshly re-randomized values and that a refreshed key is as functional as the original one.

As a consequence, even though a user at $Level_t$ has to store t values, as long as only one of them is kept in a secure place, an adversary cannot gain any advantage against the scheme: an intruder is required to break into exactly t places before a refresh is performed in order to corrupt an identity.

This natural exposure-resilience feature is a nice property of the hierarchical scheme proposed by Gentry and Silverberg and it is clearly not easy to obtain in today's certificate-based infrastructures.

3 Removing Key Escrow

It is worth noting that some people are reluctant to use IBE because of its built-in key escrow. This section describes a method recently proposed by Lee et al. ([31]) to remove the inherent key escrow from (non-hierarchical) identity based cryptography without losing the advantage of identity based public keys. The idea of this new method is to set up entities that are distinct from the PKG and whose purpose is to provide a privacy service to users' private keys by successively appending their signature on it in a blinded fashion. The system is assumed to contain n such entities called Key Privacy Authorities (KPAs) and at least one of them is assumed to remain honest throughout the system's lifetime. The key issuing protocol encompasses a setup protocol which is first jointly run by the PKG and the n KPAs:

Setup: the PKG first generates the same parameters as in the Boneh-Franklin scheme except that its public key $P_0 = s_0P$ (where s_0 is the associated private key) is no longer the overall system's public key. The n KPAs then set up their own public key $P_i = s_iP$ for $i = 1, \dots, n$ and the n PKGs jointly generate the whole system's public key: KPA₁ first computes $Y'_1 = s_1P_0$, KPA₂ then computes $Y'_2 = s_2Y'_1$ and so on. Eventually, KPA _{n} computes the system's public key $Y'_n = s_nY'_{n-1} = s_0s_1 \dots s_nP_0$. The correctness of

the process can be verified by checking that $\hat{e}(Y'_i, P) = e(Y'_{i-1}, P_i)$, where $Y'_0 = P_0$.

Key issuing: when a user of identity ID asks the PKG for a private key, he picks $x \in \mathbb{Z}_q^*$, computes a blinding factor $X = xP$ and sends it together with his identifier ID to the PKG who then computes the user's public key as $Q_{ID} = H_1(ID, P_0, P_1, \dots, P_n) \in \mathbb{G}_1$.

1. The blinded partial private key is then computed as

$$Q'_0 = H_3(\hat{e}(s_0 X, P_0)) s_0 Q_{ID}.$$

The PKG also appends a signature on Q'_0 as $Sig_0(Q'_0) = s_0 Q'_0$. The user can unblind Q'_0 since he knows x as $\hat{e}(s_0, X) = \hat{e}(P_0, P_0)^x$.

2. The user sequentially asks the n KPAs for a key privacy service. For $i = 1, \dots, n$ he sends ID, X, Q'_{i-1} and $Sig_{i-1}(Q'_{i-1})$ to KPA_i . The latter then

1. checks that $\hat{e}(Sig_{i-1}(Q'_{i-1}), P) = \hat{e}(Q'_{i-1}, P_{i-1})$,
2. computes $Q'_i = H_3(\hat{e}(s_i X, P_i)) s_i Q'_{i-1}$ and $Sig_i(Q'_i)$ that are returned by KPA_i to the user.
3. The user retrieves his private key d_{ID} by unblinding Q'_n as follows:

$$d_{ID} = \frac{Q'_n}{H_3(\hat{e}(P_0, P_0)^x) \cdots H_3(\hat{e}(P_n, P_n)^x)} = s_0 s_1 \dots s_n Q_{ID}.$$

He can then check the validity of his private key by checking that $\hat{e}(d_{ID}, P) = \hat{e}(Q_{ID}, Y)$ and use it together with $Y = s_0 s_1 \dots s_n P$ as a system's public key.

The above protocol allows removing key escrow from the Boneh-Franklin IBE as well as in other (non-hierarchical) pairing based Identity Based Signatures ([13],[27], [39],[42],...). If we compare it with other approaches ([3],[23],[25]) to avoid both digital certificates and key escrow, the present one has the clear advantage to still have human-memorizable public keys instead of unintelligible binary sequences. Furthermore, unlike what occurs in ([3]), no private channel is needed between the various entities.

The advantage over the method of Chen et al. ([14]) is that a single entity (i.e. the PKG) has to verify the user's identity before delivering him a partial private key: the n KPAs only provide private key confidentiality and do not have to perform this kind of verification. As a result, a more centralized key issuing policy can be obtained while, as in the original Boneh-Franklin scheme, a single public key (i.e. the overall system's one) still has to be certified.

4 Conclusions and Open Problems

This survey gave an overview of the solutions that identity based cryptography can provide to some of the problems encountered in PKIs: removal of digital cer-

tificates, automatic revocation and/or instantaneous revocation, natural protection against key exposure (in the case of hierarchical schemes). It also explained how the inherent key escrow (that might be an undesirable feature for many applications of asymmetric cryptography and especially for digital signatures) can be removed from any pairing based non hierarchical identity based cryptosystem without losing the advantages of identity based public keys. Among the other applications of identity based cryptosystems, we also mention that they can be turned into (strongly) key insulated cryptosystems ([5],[17],[18],[45]), forward-secure and intrusion-resilient encryption schemes (in [12] and [16], constructions are obtained from the Gentry-Silverberg hierarchical encryption scheme),

These recent advances can provide interesting alternatives to current certificate based PKIs. However, some breakthroughs still have to be achieved before obtaining fully scalable and escrow-free identity based public key infrastructures. Indeed, the key issuing protocol depicted in section 3 is convenient for non-hierarchical schemes but does not apply to each level of Gentry and Silverberg's hierarchical framework (recall that, although key escrow is somewhat restricted in their scheme, a lower-level user's private key is still known to his father). As a result, escrow-free identity based cryptography is currently only achievable in systems containing a reasonably small number of users such as local area networks (LANs). A possible direction for future research should then be to find a secure key issuing protocol for each level of hierarchical identity based schemes. Recall also, that Lee et al.'s key issuing protocol makes extensive use of bilinear maps and is thus only suitable for pairing based identity based cryptosystems. It would thus be of interest to find a counterpart of it for factoring-based or RSA-based identity based cryptosystems such as Fiat-Shamir ([20]), Guillou-Quisquater ([26]), Fischlin-Fischlin ([21]) and their variants.

Another problem that should be dealt with is the revocation problem: ID-based cryptography only provides automatic revocation of a user's public key privileges by concatenating a validity period to public identifiers (in such a way that a key is never used outside its validity period and different private keys have to be given to a user for each time period). Except a method proposed in [6] and further analyzed in [33]³ (which requires the use of an online server distinct from the PKG and holding a piece of each user's private key), no easy method to provide fine-grain revocation (which is mandatory in case of a user's key compromise) has been found so far.

Finally, another problem that is still unsolved so far is to find a practical identity based encryption scheme avoiding the use of pairings (Cocks's one is not really practical from a bandwidth point of view since a ciphertext's size is many times that of the plaintext).

³ This method is shown in [1] and [33] to also work for providing revocation in the Boneh-Franklin scheme.

References

1. J. Baek, Y. Zheng, *Identity Based Threshold Decryption*, available at <http://eprint.iacr.org/2003/164/>.
2. P.-S.-L.M. Barreto, *The Pairing Based Crypto Lounge*, web page: <http://planeta.terra.com.br/informatica/paulobarreto/pblounge.html>
3. S. Al-Riyami, K.G. Paterson, *Certificateless Public Key Cryptography*, Advances in Cryptology - Asiacrypt'03, LNCS 2894, Springer-Verlag, pp. 452-473, 2003.
4. M. Bellare, P. Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, Proc. of the 1st ACM Conference on Computer and Communications Security, pp. 62-73, 1993.
5. M. Bellare, A. Palacio, *Protecting against Key Exposure: Strongly Key-Insulated Encryption with Optimal Threshold*, available at <http://eprint.iacr.org/2002/064/>.
6. D. Boneh, X. Ding, G. Tsudik, M. Wong, *A Method for Fast Revocation of Public Key Certificates and Security Capabilities*, in proceedings of the 10th USENIX Security Symposium, pp. 297-308.
7. D. Boneh, M. Franklin, *Identity Based Encryption From the Weil Pairing*, Advances in Cryptology - Crypto'01, LNCS 2139, Springer-Verlag, pp. 213-229, 2001.
8. D. Boneh, M. Franklin, *Identity based encryption from the Weil pairing*, SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615, 2003, extended version of [7].
9. D. Boneh, C. Gentry, B. Lynn, H. Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, Advances in Cryptology - Eurocrypt'03, LNCS 2656, Springer, pp. 416-432, 2003.
10. D. Boneh, B. Lynn, H. Shacham, *Short signatures from the Weil pairing*, Advances in Cryptology - Asiacrypt'01, LNCS 2248, Springer, pp. 514-532, 2001.
11. X. Boyen. *Multipurpose identity-based signcryption: A swiss army knife for identity-based cryptography*. In *Advances in Cryptology (CRYPTO '03)*, LNCS 2729, pp. 382-398, Springer, 2003.
12. R. Canetti, S. Halevi, J. Katz, *A Forward-Secure Public-Key Encryption Scheme*, Advances in Cryptology - Eurocrypt'03, LNCS 2656, Springer, pp. 255-271, 2003.
13. J.C. Cha, J.H. Cheon, *An Identity-Based Signature from Gap Diffie-Hellman Groups*, proceedings of PKC'03. Springer, LNCS 2567, pp. 18-30, 2003.
14. L. Chen, K. Harissson, N.P. Smart, D. Soldera, *Applications of multiple trust authorities in pairing based cryptosystems*, proceedings of Infrasec 2002. Springer, LNCS 2437, pp. 260-275, 2002.
15. C. Cocks, *An Identity Based Encryption Scheme Based on Quadratic Residues*, Proc. of Cryptography and Coding, LNCS 2260, Springer, pp. 360-363, 2001.
16. Y. Dodis, M. Franklin, J. Katz, A. Miyaji, M. Yung, *Intrusion-Resilient Public-Key Encryption*, Topics in Cryptology - CT-RSA'03, LNCS 2612, Springer-Verlag, pp. 19-32, 2003.
17. Y. Dodis, J. Katz, S. Xu, M. Yung, *Key-Insulated Public Key Cryptosystems*, Advances in Cryptology - Eurocrypt'02, LNCS 2332, Springer, pp. 65-82, 2002.

18. Y. Dodis, J. Katz, S. Xu, M. Yung, *Strong Key-Insulated Signature Schemes*, Public Key Cryptography - PKC'03, LNCS 2567, Springer, pp. 130-144, 2003.
19. Y. Dodis, M. Yung, *Exposure-Resilience for Free: the Case of Hierarchical ID-based Encryption*, IEEE International Security In Storage Workshop (SISW'02), December 2002.
20. A. Fiat, A. Shamir, *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, Advances in Cryptology - Crypto'86, LNCS 0263, Springer, pp. 186-194, 1986.
21. M. Fischlin, R. Fischlin, *The Representation Problem Based on Factoring*, Topics in Cryptology - CT-RSA'02, LNCS 2271, Springer, pp. 96-113, 2002.
22. E. Fujisaki, T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes", Advances in Cryptology - Crypto'99, LNCS 1666, Springer, pp.537-554, 1999.
23. C. Gentry, *Certificate-Based Encryption and the Certificate Revocation Problem*, Advances in Cryptology - Eurocrypt'03, LNCS 2656, Springer, pp. 272-293, 2003.
24. C. Gentry, A. Silverberg, *Hierarchical ID-based cryptography*, Advances in Cryptology - Asiacrypt'02, LNCS 2501, Springer, pp. 548-566, 2002.
25. M. Girault, *Self-certified public keys*, Advances in Cryptology - Eurocrypt'91, LNCS 547, Springer, pp. 490-497, 1991.
26. L. Guillou, J.-J. Quisquater, A "Paradoxical" Identity-Based Signature Scheme Resulting From Zero-Knowledge, Advances in Cryptology - Crypto'88, LNCS 0403, Springer, pp. 216-231, 1988.
27. F. Hess, *Efficient identity based signature schemes based on pairings*, proceedings of SAC'02. Springer, LNCS 2595, pp. 310-324, 2003.
28. J. Horwitz, B. Lynn, *Toward Hierarchical Identity-Based Encryption*, Advances in Cryptology - Eurocrypt'02, LNCS 2332, Springer, pp. 466-481, 2002.
29. A. Joux, *A one round protocol for tripartite Diffie-Hellman*, Proc. of ANTS-IV, LNCS 1838, Springer, pp. 385-394, 2000.
30. A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. In *Journal of Cryptology*, volume 16-Number 4, pp. 239-247. Springer, 2003.
31. B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, S. Yoo, *Secure Key Issuing in ID-based Cryptography*, proceedings of the 2nd Australasian Information Security Workshop (AISW 2004), 2004.
32. B. Libert and J.-J. Quisquater, *New identity based signcryption schemes from pairings*, available at <http://eprint.iacr.org/2003/023>.
33. B. Libert, J.-J. Quisquater, *Efficient revocation and threshold pairing based cryptosystems*, 22nd Symposium on Principles of Distributed Computing (PODC'03), 2003.
34. J. Malone-Lee. *Identity Based Signcryption*, available at <http://eprint.iacr.org/2002/098/>.
35. A.J. Menezes, T. Okamoto, S. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Trans. on Inf. Theory, vol. 39, pp. 1639-1646, 1993.
36. A.J. Menezes, *Elliptic curve public key cryptosystems*, Kluwer Academic Publishers, 2nd printing, 1995.

37. D. Nalla, K.C. Reddy, *Signcryption scheme for Identity-based Cryptosystems*, available at <http://eprint.iacr.org/2003/066/>.
38. T. Okamoto, D. Pointcheval *The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes*, Proc. of PKC'01, LNCS 1992, Springer, pp.104-118, 2001.
39. K.G. Paterson, *ID-based signatures from pairings on elliptic curves*, available at <http://eprint.iacr.org/2002/004/>.
40. D. Pointcheval, J. Stern, *Security proofs for signature schemes*, Advances in Cryptology - Eurocrypt'96, LNCS vol. 1070, Springer, pp. 387-398, 1996.
41. D. Pointcheval, J. Stern, *Security arguments for digital signatures and blind signatures*, Journal of Cryptology, vol. 13-Number 3, pp. 361-396, 2000.
42. R. Sakai, K. Ohgishi, M. Kasahara, *Cryptosystems based on pairing*, In The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan, January 2000.
43. A. Shamir, *Identity Based Cryptosystems and Signature Schemes*, Advances in Cryptology - Crypto' 84, LNCS 0196, Springer, 1984.
44. N.P. Smart, *An identity based authenticated key agreement protocol based on the Weil pairing*, Electronic Letters, 38(13): 630-632, 2002.
45. D.-H. Yum, P.-J. Lee, *Efficient Key Updating Signature Schemes Based on IBS*, proceedings of Cryptography and Coding 2003, LNCS 2898, Springer, pp. 167-182, 2003.
46. F. Zhang, K. Kim, *ID-Based Blind Signature and Ring Signature from Pairings*. Advances in Cryptology - Asiacrypt'02, LNCS 2501, Springer, 2002.

Identity-Based Cryptography in Public Key Management

Dae Hyun Yum and Pil Joong Lee*

IS Lab., Dept. of Electronic and Electrical Eng., POSTECH, Republic of Korea
{dhyum, pj1}@postech.ac.kr

Abstract. To guarantee the authenticity of public keys, traditional PKC (Public Key Cryptography) requires certificates signed by a CA (Certification Authority). However, the management of infrastructure supporting certificates is the main complaint against traditional PKC. While identity-based PKC can eliminate this cumbersome infrastructure, the key escrow of a user's private key is inherent in identity-based PKC. Recently, new PKC paradigms were introduced: certificate-less PKC and certificate-based PKC. They retain the desirable properties of identity-based PKC without the inherent key escrow problem. A certificate-less cryptosystem eliminates the need for unwieldy certificates and a certificate-based cryptosystem simplifies the public key revocation problem. In this paper, we present an equivalence theorem among identity-based encryption, certificate-less encryption, and certificate-based encryption. We demonstrate that the three paradigms are essentially equivalent.

1 Introduction

TRADITIONAL PKC AND IDENTITY-BASED PKC. To guarantee the authenticity of a public key, the traditional PKC uses a certificate that is a digitally signed statement binding an entity and his public key. While the traditional PKC provides valuable security services such as authentication, confidentiality, integrity, and non-repudiation, the need for public key infrastructure supporting certificates is considered the main difficulty in the deployment and management of traditional PKC. In 1984, Shamir solved this problem with an identity-based PKC [16]. When Alice wants to send a message to Bob in a traditional PKC system, she must obtain Bob's authentic public key. The main idea of the identity-based PKC is to make Bob's public key derivable from some known aspect of his identity, such as an e-mail address. Hence, Alice merely derives Bob's public key directly from his identifying information when she wants to send a message in an identity-based PKC system. However, the identity-based PKC has a serious drawback in that the key escrow of the user's private key is inherent, and this

* On leave at KT Research Center.

† This research was supported by University IT Research Center Project and the Brain Korea 21 Project.

prevents the identity-based PKC from being widely adopted. Recently, new PKC paradigms were introduced: certificate-less PKC [1] and certificate-based PKC [10]. They are conceptually intermediate between traditional PKC and identity-based PKC. They enjoy the simplicity of identity-based PKC while they do not suffer from the inherent key escrow problem.

CERTIFICATE-LESS PKC. The concept of certificate-less PKC [1] grew out of a search for public key schemes that do not require the use of certificates and yet do not have the built-in key escrow feature of identity-based PKC. A certificate-less PKC system still makes use of a trusted third party (TTP) which is called the key generating center (KGC). By way of contrast to the private key generator (PKG) in identity-based PKC, the KGC does not have access to the user's private key. Instead, the KGC supplies a user with a partial private key which the KGC computes from the user's identity and a master key. The user then combines the partial private key with some secret information to generate the actual private key. The system is not identity-based, because the public key is no longer computable from a user identity. When Alice wants to send a message to Bob in a certificate-less PKC system, she must obtain Bob's public key. However, no authentication of Bob's public key is necessary and no certificate is required. The approach of certificate-less PKC looks similar to that of self-certified keys [12] and all implementations of certificate-less PKC are built from bilinear mappings on groups.

CERTIFICATE-BASED PKC. The notion of a certificate-based PKC [10] was introduced to address the problem of public key revocation. In this model, a certificate acts as a partial private key as well as a traditional public key certificate. When Bob wants to decrypt a ciphertext sent from Alice, he needs both his private key and an up-to-date certificate from the CA. Hence, senders in a certificate-based PKC system are not required to obtain fresh information on certificate status. This approach simplifies the public key revocation problem and the certificate-based PKC does not need infrastructures like CRL [13] and OCSP [14]. In a certificate-based PKC, there is no private key escrow problem, since the CA does not know the users' private keys, and no secret key distribution problem, since the CA's certificate need not be kept secret. The idea of using a certificate (or a signature from the CA) as a decryption key can be traced to Boneh-Franklin [5]. A BLS signature [8] is used as a decryption key in Boneh-Franklin. The definition of certificate-based PKC is similar to that of strongly key-insulated cryptosystems [4,9] except for the secure channel requirement. The implementation of certificate-based encryption uses a two-signer BGLS aggregate signature [7] as a decryption key.

OUR CONTRIBUTION. In this paper, we revisit the definitions and security notions of certificate-less encryption and certificate-based encryption. We provide a formal equivalence theorem among identity-based encryption, certificate-less encryption and certificate-based encryption. This shows that the three paradigms are essentially equivalent. Therefore, certificate-less encryption and certificate-based encryption can be regarded as variants of identity-based encryption.

2 Identity-Based Encryption and Certificate-Less Encryption

In this section, we review identity-based encryption schemes [5,6,11,16] and certificate-less encryption schemes [1]. After reviewing their definitions and security, we provide an equivalence result between the two encryption schemes.

2.1 Identity-Based Encryption

Definition 1. An identity-based encryption scheme is a 4-tuple of poly-time algorithms $(\text{ID_Gen}, \text{ID_Ext}, \text{ID_Enc}, \text{ID_Dec})$ such that:

- ID_Gen , the master key and parameter generation algorithm, is a probabilistic algorithm that takes as input a security parameter 1^k . It returns a master key IDSK^* and a parameter list params .
- ID_Ext , the decryption key issuance algorithm, is a deterministic algorithm that takes as input a user identity id , a parameter list params , and a master key IDSK^* . It returns the user id 's decryption key IDSK_{id} .
- ID_Enc , the encryption algorithm, is a probabilistic algorithm that takes as input a message M , a user identity id , and a parameter list params . $\text{ID_Enc}_{\text{params}}(M, id)$ returns a ciphertext C .
- ID_Dec , the decryption algorithm, is a deterministic algorithm that takes as input a parameter list params , the decryption key IDSK_{id} , and a ciphertext C . $\text{ID_Dec}_{\text{params}}^{\text{IDSK}_{id}}(C)$ returns a message M or the special symbol \perp .

We require that for all message M , $\text{ID_Dec}_{\text{params}}^{\text{IDSK}_{id}}(\text{ID_Enc}_{\text{params}}(M, id)) = M$.

In an identity-based encryption scheme, ID_Gen and ID_Ext are performed by a PKG. A decryption key IDSK_{id} is given to a user id by the PKG through a secure channel. Note that the key escrow of the user's private key is inherent in an identity-based encryption scheme.

For security analysis, we define a key exposure oracle $\text{ID_Exp}_{\text{params}}^{\text{IDSK}^*}(\cdot)$ that returns a decryption key IDSK_{id} on input id . We also give the adversary access to a decryption oracle $\text{ID_Dec}_{\text{params}}^{\text{IDSK}^*}(\cdot, \cdot)$ that returns $\text{ID_Dec}_{\text{params}}^{\text{IDSK}_{id}}(C)$ on input (id, C) . Finally, the adversary can access a left-or-right encryption oracle $\text{ID_Enc}_{\text{params}}(\cdot, \text{LR}(\cdot, \cdot, b))$ that given id_{ch} and equal length messages M_0, M_1 returns a challenge ciphertext $C_{ch} = \text{ID_Enc}_{\text{params}}(M_b, id_{ch})$ [3].

The security goal of an identity-based encryption scheme is chosen ciphertext security. This means that any probabilistic polynomial time (PPT) adversary A should have a negligible advantage of distinguishing the encryptions of two messages of his choice given access to the key exposure oracle $\text{ID_Exp}_{\text{params}}^{\text{IDSK}^*}(\cdot)$, the decryption oracle $\text{ID_Dec}_{\text{params}}^{\text{IDSK}^*}(\cdot, \cdot)$, and the left-or-right encryption oracle $\text{ID_Enc}_{\text{params}}(\cdot, \text{LR}(\cdot, \cdot, b))$. The key exposure oracle models the ability of the adversary to compromise any user of his choice, except the target user.

Definition 2. Let Π_{ID} be an identity-based encryption scheme. For any adversary A , we may define the following:

$$\begin{aligned} \text{Succ}_{A, \Pi_{ID}}(k) = \\ \Pr[b' = b : (\text{IDSK}^*, \text{params}) \leftarrow \text{ID_Gen}(1^k); b \leftarrow \{0, 1\}; \\ b' \leftarrow A^{\text{ID_Exp}_{\text{params}}^{\text{IDSK}^*}(\cdot), \text{ID_Dec}_{\text{params}}^{\text{IDSK}^*}(\cdot, \cdot), \text{ID_Enc}_{\text{params}}(\cdot, \text{LR}(\cdot, \cdot, b))}(\text{params})] \end{aligned}$$

where the adversary may query oracles adaptively subject to the restriction that it can make exactly one query to the left-or-right encryption oracle. Let id_{ch} be the user identity of the query to the left-or-right encryption oracle and C_{ch} be the challenge ciphertext returned by the left-or-right encryption oracle. We say that A succeeds if $b' = b$, id_{ch} was never submitted to the key exposure oracle, and (id_{ch}, C_{ch}) was never submitted to the decryption oracle after C_{ch} was returned by the left-or-right encryption oracle. Π_{ID} is said to be secure against chosen-ciphertext attacks if for any PPT A , the advantage $\text{Adv}_{A, \Pi_{ID}}(k) = 2 \times |\text{Succ}_{A, \Pi_{ID}}(k) - 1/2|$ is negligible.

2.2 Certificate-Less Encryption

Definition 3. A certificate-less encryption scheme is a 7-tuple of poly-time algorithms $(\text{CL_Gen}, \text{CL_Ext_Partial_Pri_Key}, \text{CL_Set_Sec_Val}, \text{CL_Set_Pri_Key}, \text{CL_Set_Pub_Key}, \text{CL_Enc}, \text{CL_Dec})$ such that:

- CL_Gen , the master key and parameter generation algorithm, is a probabilistic algorithm that takes as input a security parameter 1^k . It returns a master key CLSK^* and a parameter list params .
- $\text{CL_Ext_Partial_Pri_Key}$, the partial private key issuance algorithm, is a deterministic algorithm that takes as input id , params , and CLSK^* . It returns the user id 's partial private key CLD_{id} .
- CL_Set_Sec_Val , the secret value setup algorithm, is a probabilistic algorithm that takes as input params and id . It returns the user id 's secret value CLS_{id} .
- CL_Set_Pri_Key , the private key generation algorithm, is a deterministic algorithm that takes as input params , CLD_{id} , and CLS_{id} . It returns the user id 's private key CLSK_{id} .
- CL_Set_Pub_Key , the public key generation algorithm, is a deterministic algorithm that takes as input params , id , and CLS_{id} . It returns the user id 's public key CLPK_{id} .
- CL_Enc , the encryption algorithm, is a probabilistic algorithm that takes as input a message M , id , params , and CLPK_{id} . $\text{CL_Enc}_{\text{params}}^{\text{CLPK}_{\text{id}}}(M, \text{id})$ returns a ciphertext C .
- CL_Dec , the decryption algorithm, is a deterministic algorithm that takes as input params , CLSK_{id} , and C . $\text{CL_Dec}_{\text{params}}^{\text{CLSK}_{\text{id}}}(C)$ returns M or \perp .

We require that for all message M , $\text{CL_Dec}_{\text{params}}^{\text{CLSK}_{\text{id}}}(\text{CL_Enc}_{\text{params}}^{\text{CLPK}_{\text{id}}}(M, \text{id})) = M$.

In a certificate-less encryption scheme, CL_Gen and $\text{CL_Ext_Partial_Pri_Key}$ are performed by a KGC. A partial private key CLD_{id} is given to a user id by the KGC through a secure channel. Since CL_Set_Sec_Val , CL_Set_Pri_Key , and CL_Set_Pub_Key are executed by a user, the key escrow of the user's private key is not inherent in a certificate-less encryption scheme.

For security considerations, we extend the model of an identity-based encryption scheme to allow an adversary to extract partial private keys, or private keys, or both, for identities of his choice. We must also consider the ability of the adversary to replace the public key of any entity with a value of his choice, because there is no certificate in a certificate-less encryption scheme. Six oracles can be accessed by the adversary. The first is a partial private key exposure oracle $\text{CL_Exp_Partial}_{params}^{CLSK^*}(\cdot)$ that returns CLD_{id} on input a user identity id . The second is a private key exposure oracle $\text{CL_Exp_Pri}_{params}^{CLSK^*}(\cdot)$ that returns $CLSK_{id}$ on input a user identity id if id 's public key has not been replaced. The third is a public key broadcast oracle $\text{CL_Bro_Pub}_{params}(\cdot)$ that returns $CLPK_{id}$ on input a user identity id . The fourth is a public key replacement oracle $\text{CL_Rep_Pub}_{params}(\cdot, \cdot)$ that replaces the public key $CLPK_{id}$ for a user id with $CLPK'_{id}$ on input $(id, CLPK'_{id})$. The fifth is a decryption oracle $\text{CL_Dec}_{params}^{CLSK^*}(\cdot, \cdot)$ that returns $\text{CL_Dec}_{params}^{CLSK_{id}}(C)$ on input (id, C) . The sixth is a left-or-right encryption oracle $\text{CL_Enc}_{params}(\cdot, LR(\cdot, \cdot, b))$ that given a user identity id_{ch} and equal length messages M_0, M_1 returns a challenge ciphertext $C_{ch} = \text{CL_Enc}_{params}^{CLPK_{id}}(M_b, id_{ch})$.

The security of a certificate-less encryption scheme is against two different types of adversaries. The Type I adversary A_I has no access to the master key, but may replace public keys, extract partial private and private keys, and make decryption queries. The Type II adversary A_{II} equipped with the master key models an eavesdropping KGC. A_{II} is not allowed to replace public keys. For detailed restrictions on the adversaries, refer to [1] and Appendix A.

Definition 4. Let Π_{CL} be a certificate-less encryption scheme. For any adversary A , we may define the following:

$$\begin{aligned} \text{Succ}_{A, \Pi_{CL}}(k) = \\ Pr[b' = b : (CLSK^*, params) \leftarrow \text{CL_Gen}(1^k); b \leftarrow \{0, 1\}; \\ b' \leftarrow A^{O_1(\cdot), O_2(\cdot), O_3(\cdot), \text{CL_Exp_Pri}_{params}^{CLSK^*}(\cdot), \text{CL_Bro_Pub}_{params}(\cdot), \text{CL_Dec}_{params}^{CLSK^*}(\cdot)}(params, h)] \end{aligned}$$

where $h = \perp$, $O_1(\cdot) = \text{CL_Exp_Partial}_{params}^{CLSK^*}(\cdot)$, $O_2(\cdot) = \text{CL_Rep_Pub}_{params}(\cdot, \cdot)$, $O_3 = \text{CL_Enc}_{params}(\cdot, LR(\cdot, \cdot, b))$ for A_I , and $h = CLSK^*$, $O_1(\cdot) = \perp$, $O_2(\cdot) = \perp$, $O_3 = \text{CL_Enc}_{params}(\cdot, LR(\cdot, \cdot, b))$ for A_{II} . The adversary may query oracles adaptively except that it can make exactly one query to the left-or-right encryption oracle. A must follow the adversarial constraints given in [1]. Π_{CL} is said to be secure against chosen-ciphertext attacks if for any PPT A , the advantage $\text{Adv}_{A, \Pi_{CL}}(k) = 2 \times |\text{Succ}_{A, \Pi_{CL}}(k) - 1/2|$ is negligible.

2.3 Equivalence Result

Certificate-less encryption is conceptually intermediate between traditional public key encryption and identity-based encryption. Here, we show that certificate-less encryption is closer (actually, equivalent) to identity-based encryption. In light of this theorem, it is natural that the implementation of certificate-less encryption [1] is based on Boneh-Franklin identity-based encryption [5].

Theorem 1. *A secure certificate-less encryption scheme exists if and only if there is a secure identity-based encryption scheme.*

Proof. Let $\Pi_{ID} = (\text{ID_Gen}, \text{ID_Ext}, \text{ID_Enc}, \text{ID_Dec})$ be a secure identity-based encryption scheme. From Π_{ID} , we build a secure certificate-less encryption scheme $\Psi_{CL} = (\text{CL_Gen}, \text{CL_Ext_Partial_Pri_Key}, \text{CL_Set_Sec_Val}, \text{CL_Set_Pri_Key}, \text{CL_Set_Pub_Key}, \text{CL_Enc}, \text{CL_Dec})$. The underlying idea is that different implementations of Π_{ID} are used by the KGC and users for the construction of Ψ_{CL} to overcome the private key escrow property of Π_{ID} . The security for Ψ_{CL} is proved by Lemma 1 in Appendix B.

$\text{CL_Gen}(1^k)$ $(IDSK_{KGC}^*, params_{KGC}) \leftarrow \text{ID_Gen}(1^k);$ $CLSK^* \leftarrow IDSK_{KGC}^*;$ $params \leftarrow params_{KGC};$ Return $(CLSK^*, params)$	$\text{CL_Set_Pub_Key}(params, id, CLS_{id})$ Parse CLS_{id} as $(IDSK_{id}^\#, params_{id});$ $CLPK_{id} \leftarrow params_{id};$ Return $CLPK_{id}$
$\text{CL_Ext_Partial_Pri_Key}(id, params, CLSK^*)$ $IDSK_{id}^\alpha \leftarrow \text{ID_Ext}(id, params, CLSK^*);$ $CLD_{id} \leftarrow IDSK_{id}^\alpha;$ Return CLD_{id}	$\text{CL_Enc}(M, id, params, CLPK_{id})$ $params_{id} \leftarrow CLPK_{id};$ $C' \leftarrow \text{ID_Enc}_{params_{id}}(M, id);$ $C \leftarrow \text{ID_Enc}_{params}(C', id);$ Return C
$\text{CL_Set_Sec_Val}(params, id)$ $(IDSK_{id}^\#, params_{id}) \leftarrow \text{ID_Gen}(1^k);$ $CLS_{id} \leftarrow (IDSK_{id}^\#, params_{id});$ Return CLS_{id}	$\text{CL_Dec}(params, CLSK_{id}, C)$ Parse $CLSK_{id}$ as $(CLD_{id}, IDSK_{id}^\beta, params_{id});$ $C' \leftarrow \text{ID_Dec}_{params_{id}}^{CLD_{id}}(C);$ $M \leftarrow \text{ID_Dec}_{params_{id}}^{IDSK_{id}^\beta}(C');$ Return M
$\text{CL_Set_Pri_Key}(params, CLD_{id}, CLS_{id})$ Parse CLS_{id} as $(IDSK_{id}^\#, params_{id});$ $IDSK_{id}^\beta \leftarrow \text{ID_Ext}(id, params_{id}, IDSK_{id}^\#);$ $CLSK_{id} \leftarrow (CLD_{id}, IDSK_{id}^\beta, params_{id});$ Return $CLSK_{id}$	

Conversely, let $\Pi_{CL} = (\text{CL_Gen}, \text{CL_Ext_Partial_Pri_Key}, \text{CL_Set_Sec_Val}, \text{CL_Set_Pri_Key}, \text{CL_Set_Pub_Key}, \text{CL_Enc}, \text{CL_Dec})$ be a secure certificate-less encryption scheme. From Π_{CL} , we build a secure identity-based encryption scheme $\Psi_{ID} = (\text{ID_Gen}, \text{ID_Ext}, \text{ID_Enc}, \text{ID_Dec})$. The idea is that a common secret value $CLSPKG$ is used for all users. Note that the exposure of CLS_{id} alone reveals no information on the user id 's private key in a secure certificate-less encryption scheme, since the input of CL_Set_Pub_Key is $(params, id, CLS_{id})$ and the partial private key CLD_{id} can be chosen independently of the public key.

The security of derived encryption scheme Ψ_{ID} can be proved by the security of the underlying encryption scheme Π_{CL} . See Lemma 2 in Appendix B. \square

$\text{ID_Gen}(1^k)$ $(\text{CLSK}^*, \text{params}_{cl}) \leftarrow \text{CL_Gen}(1^k);$ CLSPKG $\leftarrow \text{CL_Set_Sec_Val}(\text{params}_{cl}, \text{PKG});$ $\text{IDSK}^* \leftarrow \text{CLSK}^*;$ $\text{params} \leftarrow (\text{params}_{cl}, \text{CLSPKG});$ Return $(\text{IDSK}^*, \text{params})$	$\text{ID_Ext}(id, \text{params}, \text{IDSK}^*)$ Parse params as $(\text{params}_{cl}, \text{CLSPKG});$ CLD_{id} $\leftarrow \text{CL_Ext_Partial_Pri_Key}(id, \text{params}_{cl}, \text{IDSK}^*);$ CLSK_{id} $\leftarrow \text{CL_Set_Pri_Key}(\text{params}_{cl}, \text{CLD}_{id}, \text{CLSPKG});$ $\text{IDSK}_{id} \leftarrow \text{CLSK}_{id};$ Return IDSK_{id}
$\text{ID_Enc}(M, id, \text{params})$ Parse params as $(\text{params}_{cl}, \text{CLSPKG});$ CLPK_{id} $\leftarrow \text{CL_Set_Pub_Key}(\text{params}_{cl}, id, \text{CLSPKG});$ $C \leftarrow \text{CL_Enc}(M, id, \text{params}_{cl}, \text{CLPK}_{id});$ Return C	$\text{ID_Dec}(\text{params}, \text{IDSK}_{id}, C)$ Parse params as $(\text{params}_{cl}, \text{CLSPKG});$ $M \leftarrow \text{CL_Dec}_{\text{params}_{cl}}^{\text{IDSK}_{id}}(C);$ Return M

3 Certificate-Based Encryption and Equivalence Result

In this section, we review certificate-based encryption schemes [10] and provide an equivalence result between certificate-based encryption and identity-based encryption. Along with Theorem 1, we can reach the conclusion that the following paradigms are essentially equivalent: identity-based encryption, certificate-less encryption, and certificate-based encryption.

3.1 Certificate-Based Encryption

Definition 5. A certificate-based encryption scheme is a 6-tuple of poly-time algorithms $(\text{CB_Gen}, \text{CB_Set_User_Key}, \text{CB_Upd}^{\text{CA}}, \text{CB_Upd}^{\text{User}}, \text{CB_Enc}, \text{CB_Dec})$ such that:

- CB_Gen , the master key and parameter generation algorithm, is a probabilistic algorithm that takes as input a security parameter 1^k and (optionally) the total number of time periods T . It returns a master key CBSK^* and a parameter list params .
- CB_Set_User_Key , the user key generation algorithm, is a probabilistic algorithm that takes as input params , id , and (optionally) T . It returns the user id 's private key CBSK_{id} and public key CBPK_{id} .
- $\text{CB_Upd}^{\text{CA}}$, the CA update algorithm, is a deterministic algorithm that takes as input params , id , a time period t , CBPK_{id} , and CBSK^* . It returns the user id 's certificate $\text{Cert}'_{(id,t)}$ for the time period t .
- $\text{CB_Upd}^{\text{User}}$, the user update algorithm, is a deterministic algorithm that takes as input params , t , $\text{Cert}'_{(id,t)}$ and (optionally) $\text{Cert}_{(id,t-1)}$. It returns $\text{Cert}_{(id,t)}$.
- CB_Enc , the encryption algorithm, is a probabilistic algorithm that takes as input M , id , params , t and CBPK_{id} . $\text{CB_Enc}_{\text{params}}^{\text{CBPK}_{id}}(M, id, t)$ returns a ciphertext C .

- **CB_Dec**, the decryption algorithm, is a deterministic algorithm that takes as input $params$, CB_{SK}_{id} , C , and $Cert_{(id,t)}$. $CB_Dec_{params}^{CB_{SK}_{id}}(C, Cert_{(id,t)})$ returns a message M or \perp .

For every message M , $CB_Dec_{params}^{CB_{SK}_{id}}(CB_Enc_{params}^{CB_{PK}_{id}}(M, id, t), Cert_{(id,t)}) = M$.

In certificate-based encryption, **CB_Gen** and **CB_Upd^{CA}** are performed by a CA. A certificate $Cert'_{(id,t)}$ is given to a user id by the CA not necessarily through a secure channel. Since **CB_Set_User_Key** is executed by a user, the key escrow of the user's private key is not inherent in a certificate-based encryption scheme.

For security analysis, we give the adversary access to four types of oracles. The first is a certification oracle $CB_Upd_{params}^{CB_{SK}^*}(\cdot, \cdot, \cdot)$ that returns $Cert'_{(id,t)}$ on input a user identity id , a time period t , and the user id 's public key CB_{PK}_{id} . The second is a public key broadcast oracle $CB_Bro_Pub_{params}(\cdot)$ that returns CB_{PK}_{id} on input a user identity id . The third is a decryption oracleⁱ $CB_Dec_{params}^{CB_{SK}^*}(\cdot, \cdot, \cdot, \cdot)$ that returns $CB_Dec_{params}^{CB_{SK}_{id}}(C, Cert_{(id,t)})$ on input (id, t, C, CB_{PK}_{id}) . The fourth is a left-or-right encryption oracle $CB_Enc_{params}(\cdot, \cdot, LR(\cdot, \cdot, b))$ that given id_{ch} , a time period t , and equal length messages M_0, M_1 returns a challenge ciphertext $C_{ch} = CB_Enc_{params}^{CB_{PK}_{id}}(M_b, id_{ch}, t)$.

The security of a certificate-based encryption scheme is against two different types of adversaries. The Type I adversary A_I has no access to the master key, but may make certification queries and decryption queries. The Type II adversary A_{II} equipped with the master key models an eavesdropping CA. For detailed restrictions on the two types of adversaries and security notions, refer to [10] and Appendix A.

Definition 6. Let Π_{CB} be a certificate-based encryption scheme. For any adversary A , we may define the following:

$$\begin{aligned} & Succ_{A, \Pi_{CB}}(k) = \\ & Pr[b' = b : (CB_{SK}^*, params) \leftarrow CB_Gen(1^k); b \leftarrow \{0, 1\}; \\ & b' \leftarrow A^{O(\cdot), CB_Bro_Pub_{params}(\cdot), CB_Dec_{params}^{CB_{SK}^*}(\cdot), CB_Enc_{params}(\cdot, \cdot, LR(\cdot, \cdot, b))}(params, h)] \end{aligned}$$

where $h = \perp$, $O(\cdot) = CB_Upd_{params}^{CB_{SK}^*}(\cdot)$ for A_I , and $h = CB_{SK}^*$, $O(\cdot) = \perp$ for A_{II} . The adversary may query oracles adaptively, except that it can make exactly one query to the left-or-right encryption oracle. The adversary must follow the adversarial constraints given in [10]. Π_{CB} is said to be secure against chosen-ciphertext attacks if for any PPT A , the advantage $Adv_{A, \Pi_{CB}}(k) = 2 \times |Succ_{A, \Pi_{CB}}(k) - 1/2|$ is negligible.

3.2 Equivalence Result

Theorem 2. A secure certificate-based encryption scheme exists if and only if there is a secure identity-based encryption scheme.

ⁱ If the input CB_{PK}_{id} is inconsistent with the registered public key, the decryption oracle returns \perp . Otherwise, the decryption oracle's answer is assumed to be correct.

Proof. The proof strategy for Theorem 2 is similar to that for Theorem 1. One noteworthy point is that identity-based encryption requires a secure channel between the PKG and a user, while certificate-based encryption does not. However, this difference should be handled by external mechanisms rather than definitional algorithms. Let $\Pi_{ID} = (\text{ID_Gen}, \text{ID_Ext}, \text{ID_Enc}, \text{ID_Dec})$ be a secure identity-based encryption scheme. From Π_{ID} , we build a secure certificate-based encryption scheme $\Gamma_{CB} = (\text{CB_Gen}, \text{CB_Set_User_Key}, \text{CB_Upd}^{\text{CA}}, \text{CB_Upd}^{\text{User}}, \text{CB_Enc}, \text{CB_Dec})$. The optional inputs T and $\text{Cert}_{(id,t-1)}$ of CB_Gen , CB_Set_User_Key and $\text{CB_Upd}^{\text{User}}$ are ignored for simplicity. We assume that the security parameter k can be derived from $params$, since $params$ is derived from the security parameter k and k can be included in $params$. The security of Γ_{CB} can be proved by the security of the Π_{ID} in a similar manner to Lemma 1.

$\text{CB_Gen}(1^k)$ $(\text{IDSK}_{CA}^*, \text{params}_{CA}) \leftarrow \text{ID_Gen}(1^k);$ $\text{CBSK}^* \leftarrow \text{IDSK}_{CA}^*;$ $\text{params} \leftarrow \text{params}_{CA};$ Return $(\text{CBSK}^*, \text{params})$	$\text{CB_Upd}^{\text{CA}}(\text{params}, id, t, \text{CBPK}_{id}, \text{CBSK}^*)$ $\text{IDSK}_{(id,t)}^\alpha \leftarrow \text{ID_Ext}((id, t, \text{CBPK}_{id}), \text{params}, \text{CBSK}^*);$ $\text{Cert}'_{(id,t)} \leftarrow \text{IDSK}_{(id,t)}^\alpha;$ Return $\text{Cert}'_{(id,t)}$
$\text{CB_Set_User_Key}(\text{params}, id)$ $(\text{IDSK}_{id}^\#, \text{params}_{id}) \leftarrow \text{ID_Gen}(1^k);$ $\text{IDSK}_{id}^\beta \leftarrow \text{ID_Ext}(id, \text{params}_{id}, \text{IDSK}_{id}^\#);$ $\text{CBSK}_{id} \leftarrow (\text{IDSK}_{id}^\beta, \text{params}_{id});$ $\text{CBPK}_{id} \leftarrow \text{params}_{id};$ Return $(\text{CBSK}_{id}, \text{CBPK}_{id})$	$\text{CB_Upd}^{\text{User}}(\text{params}, t, \text{Cert}'_{(id,t)})$ $\text{Cert}_{(id,t)} \leftarrow \text{Cert}'_{(id,t)};$ Return $\text{Cert}_{(id,t)}$
$\text{CB_Enc}(M, id, \text{params}, t, \text{CBPK}_{id})$ $C' \leftarrow \text{ID_Enc}_{\text{CBPK}_{id}}(M, id);$ $C \leftarrow \text{ID_Enc}_{\text{params}}(C', (id, t, \text{CBPK}_{id}));$ Return C	$\text{CB_Dec}(\text{params}, \text{CBSK}_{id}, C, \text{Cert}_{(id,t)})$ Parse CBSK_{id} as $(\text{IDSK}_{id}^\beta, \text{params}_{id});$ $C' \leftarrow \text{ID_Dec}_{\text{params}_{id}}^{\text{Cert}_{(id,t)}}(C);$ $M \leftarrow \text{ID_Dec}_{\text{params}_{id}}^{\text{IDSK}_{id}^\beta}(C');$ Return M

Conversely, let $\Pi_{CB} = (\text{CB_Gen}, \text{CB_Set_User_Key}, \text{CB_Upd}^{\text{CA}}, \text{CB_Upd}^{\text{User}}, \text{CB_Enc}, \text{CB_Dec})$ be a secure certificate-based encryption scheme. From Π_{CB} , we build a secure identity-based encryption scheme $\Gamma_{ID} = (\text{ID_Gen}, \text{ID_Ext}, \text{ID_Enc}, \text{ID_Dec})$. The underlying idea is that common private key CBSK_{PKG} and public key CBPK_{PKG} are used for all users. In addition, we fix the time period as t_{PKG} to nullify the certificate updating procedure.

The security of derived encryption scheme Γ_{ID} can be proved by the security of the underlying encryption scheme Π_{CB} in a similar manner to Lemma 2. \square

Finally, from Theorem 1 and Theorem 2, we can obtain Theorem 3, the equivalence result among three paradigms: identity-based encryption, certificate-less encryption, and certificate-based encryption.

```

ID_Gen( $1^k$ )
  ( $CB_{SK}^*, params_{cb}$ )  $\leftarrow$  CB_Gen( $1^k$ );
  ( $CB_{SK}_{PKG}, CB_{PK}_{PKG}$ )  $\leftarrow$  CB_Set_User_Key( $params_{cb}, PKG$ );
   $params \leftarrow (params_{cb}, CB_{SK}_{PKG}, CB_{PK}_{PKG})$ ;
   $ID_{SK}^* \leftarrow CB_{SK}^*$ ;
  Return ( $ID_{SK}^*, params$ )

ID_Ext( $id, params, ID_{SK}^*$ )
  Parse  $params$  as ( $params_{cb}, CB_{SK}_{PKG}, CB_{PK}_{PKG}$ );
   $Cert'_{(id, t_{PKG})} \leftarrow$  CB_UpdCA( $params_{cb}, id, t_{PKG}, CB_{PK}_{PKG}, ID_{SK}^*$ );
   $Cert_{(id, t_{PKG})} \leftarrow$  CB_UpdUser( $params_{cb}, t_{PKG}, Cert'_{(id, t_{PKG})}$ );
   $ID_{SK}_{id} \leftarrow Cert_{(id, t_{PKG})}$ ;
  Return  $ID_{SK}_{id}$ 

ID_Enc( $M, id, params$ )
  Parse  $params$  as ( $params_{cb}, CB_{SK}_{PKG}, CB_{PK}_{PKG}$ );
   $C \leftarrow$  CB_Enc( $M, id, params_{cb}, t_{PKG}, CB_{PK}_{PKG}$ );
  Return  $C$ 

ID_Dec( $params, ID_{SK}_{id}, C$ )
  Parse  $params$  as ( $params_{cb}, CB_{SK}_{PKG}, CB_{PK}_{PKG}$ );
   $M \leftarrow$  CB_Dec( $params_{cb}, CB_{SK}_{PKG}, C, ID_{SK}_{id}$ );
  Return  $M$ 

```

Theorem 3. *The existence of one of the following encryption schemes entails the other two encryption schemes: a secure identity-based encryption scheme, a secure certificate-less encryption scheme, and a secure certificate-based encryption scheme.*

4 Concluding Remarks

After the first practical and secure identity-based encryption scheme was presented in [5], the use of identity-based schemes to achieve added security became quite popular. In this paper, we showed that certificate-less encryption, certificate-based encryption, and identity-based encryption are all equivalent. Our equivalence theorem reveals that the implications of Shamir's identity-based cryptography are richer than first observed. We expect that this exploration will continue.

References

1. S. S. Al-Riyami and K. G. Peterson, "Certificateless public key cryptography," *Asiacrypt 2003*, LNCS Vol. 2894, pp. 452-473, 2003.
2. M. Bellare and P. Rogaway, "Random Oracles are Practical: a Paradigm for Designing Efficient Protocols," *1st ACM Conf. on Computer and Communications Security*, pp. 62-73, 1993.

3. M. Bellare, A. Desai, D. Jorjipii, and P. Rogaway, "A concrete security treatment of symmetric encryption: analysis of the DES modes of operation," FOCS 1997, IEEE, 1997.
4. M. Bellare and A. Palacio, "Protecting against key exposure: strong key-insulated encryption with optimal threshold," Cryptology ePrint archive 2002/064.
5. D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," Crypto 2001, LNCS Vol. 2139, pp. 213-229, 2001.
6. D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615, 2003.
7. D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," Eurocrypt 2003, LNCS Vol. 2656, pp. 416-432, 2003.
8. D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," Asiacrypt 2001, LNCS Vol. 2248, pp. 514-532, 2001.
9. Y. Dolis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," Eurocrypt 2002, LNCS Vol. 2332, pp. 65-82, 2002.
10. C. Gentry, "Certificate-based encryption and the certificate revocation problem," Eurocrypt 2003, LNCS Vol. 2656, pp. 272-293, 2003.
11. C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography," Asiacrypt 2002, LNCS Vol. 2501, pp. 548-566, 2002.
12. M. Girault, "Self-certified public keys," Eurocrypt 1991, LNCS Vol. 547, pp. 490-497, 1992.
13. R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," RFC 3280, IETF, 2002.
14. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 internet public key infrastructure online certificate status protocol - OCSP," RFC 2560, IETF, 1999.
15. C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," Crypto 1991, LNCS Vol. 547, pp. 433-444, 1991.
16. A. Shamir, "Identity-based cryptosystems and signature schemes," Crypto 1984, LNCS Vol. 196, pp. 47-53, 1984.

Appendix A. On the Security Notions

We consider some differences in the definition and security for certificate-less encryption [1] and certificate-based encryption [10], in brief.

In certificate-less encryption, the adversary can replace the public key of any user with a value of his choice. This stems from the elimination of certificates. While this ability of the adversary is modeled as access to the public key replacement oracle $\text{CL_Rep_Pub}_{\text{params}}(\cdot, \cdot)$, there is no such oracle in certificate-based encryption. However, the inputs of the decryption oracle $\text{CB_Dec}_{\text{params}}^{\text{CBSK}^*}(\cdot, \cdot, \cdot, \cdot)$ and the certification oracle $\text{CB_Upd}_{\text{params}}^{\text{CBSK}^*}(\cdot, \cdot, \cdot)$ include a public key CBPK_{id} which is omitted in the input of $\text{CL_Dec}_{\text{params}}^{\text{CLSK}^*}(\cdot, \cdot)$. Hence, the adversary in certificate-based encryption can simulate the public key replacement attack by giving a fake public key CBPK'_{id} to the decryption oracle $\text{CB_Dec}_{\text{params}}^{\text{CBSK}^*}(\cdot, \cdot, \cdot, \cdot)$ and the certification oracle $\text{CB_Upd}_{\text{params}}^{\text{CBSK}^*}(\cdot, \cdot, \cdot)$.

The security of a certificate-less encryption scheme is against two different types of adversaries. The Type I adversary A_I has no access to the master key, but may replace public keys, extract partial private and private keys, and make decryption queries. When A_I has replaced the public key of a user id and asks the decryption oracle about the id 's ciphertexts, the decryption oracle can take two options: (1) the decryption oracle should somehow properly decrypt ciphertexts for users whose public keys have been replaced, or (2) the decryption oracle could output decryptions which might be incorrect or meaningless. The first option was adopted in certificate-less encryption and the second option in certificate-based encryption. We adopt the latter behavior of the decryption oracle, because we will construct certificate-less encryption and certificate-based encryption based on general primitives and do without any specific assumptions such as the random oracle model [2]. However, we assume that the ciphertext can be decrypted properly, if the replaced public key ($CLPK_{id}$) and the corresponding secret information (CLS_{id} or $CLSK_{id}$) are additionally submitted to the decryption oracle. This behavior of the decryption oracle with the optional input also holds for the certificate-based encryption oracle. The certificate-less attacker A_I is not allowed to submit id_{ch} to the partial private key exposure oracle; $CLD_{id_{ch}}$ is securely given to the user id_{ch} by definition and can be deleted after generating the public/private key pair. Note that A_I can replace the public key of id_{ch} . The exposure of $CLD_{id_{ch}}$ can be treated by the Type II adversary A_{II} equipped with the master key. A_{II} models an eavesdropping KGC and generates partial private keys by himself. A_{II} is not allowed to replace public keys.

Appendix B. Security Proofs

Lemma 1. Ψ_{CL} is a secure certificate-less encryption scheme if Π_{ID} is secure against chosen-ciphertext attacks.

Proof. (Sketch)ⁱⁱ Let A_I be a Type I attacker who can break Ψ_{CL} . Suppose that A_I has advantage ϵ and runs in time t . We show how to construct from A_I an adversary A' against Π_{ID} . At the beginning, A' is given by a Π_{ID} challenger a parameter list $params$ and three Π_{ID} oracles. To run A_I , A' simulates the $CL_Gen(1^k)$ by supplying A_I with $params$. A' keeps a list $L = \{(id, IDSK_{id}^\alpha, params_{id}, IDSK_{id}^\beta)\}$ where $IDSK_{id}^\alpha$ is the output of key exposure oracle $ID_Exp_{params}^{IDSK^*}(\cdot)$. The list L is computed according to the A_I 's queries. A' responds to A_I 's Ψ_{CL} oracle queries with the list L and three Π_{ID} oracles. When the Type I attacker A_I outputs b' , A' outputs the same b' to the Π_{ID} challenger. Since the A_I 's view is identical to its view in the real attack, $\text{Adv}_{A', \Pi_{ID}}(k) = 2 \times |\Pr(b' = b) - 1/2| \geq \epsilon$ and A' runs in time $O(\text{time}(t))$.

Let A_{II} be a Type II attacker who can break Ψ_{CL} . Suppose that A_{II} has advantage ϵ , runs in time t , and makes queries on l users, i.e., $(id_1, id_2, \dots,$

ⁱⁱ The original proof of Lemma 1 was somewhat long (about 4~5 pages). However, only outline is presented here for space limitation. Please refer to the full version of this paper for detailed proof.

id_l). We show how to construct from A_{II} an adversary A'' against Π_{ID} . At the beginning, A'' is given by a Π_{ID} challenger a parameter list $params_{ch}$ and three Π_{ID} oracles. To simulate $\text{CL_Gen}(1^k)$, A'' runs $\text{ID_Gen}(1^k)$ to obtain $(IDSK_{KGC}^*, params_{KGC})$ and sets $params = params_{KGC}$, $CLSK^* = IDSK_{KGC}^*$. A'' gives $(params, CLSK^*)$ to A_{II} since A_{II} has access to the master key. As before, A'' keeps a list $L = \{(id, IDSK_{id}^\alpha, params_{id}, IDSK_{id}^\beta)\}$ where $IDSK_{id}^\alpha$ is the output of $\text{ID_Ext}(id, params, IDSK_{KGC}^*)$. A'' chooses a random index $j \in \{1, \dots, l\}$ and sets $IDSK_{id_j}^\alpha = \text{ID_Ext}(id_j, params, IDSK_{KGC}^*)$, $params_{id_j} = params_{ch}$, $IDSK_{id_j}^\beta = \perp$. The element $(id_j, IDSK_{id_j}^\alpha, params_{id_j}, IDSK_{id_j}^\beta)$ is added to the list L . The remainder of the list L is computed according to the A_{II} 's queries. A'' responds to A_{II} 's Ψ_{CL} oracle queries with the list L and three Π_{ID} oracles. When the Type II attacker A_{II} outputs b' , A'' outputs the same b' to the Π_{ID} challenger. If A'' does not abort during the simulation, the A_{II} 's view is identical to its view in the real attack. Since the index j is chosen randomly, the probability that A'' does not abort during the simulation is $1/l$. Hence, the advantage of A'' satisfies $\text{Adv}_{A'', \Pi_{ID}}(k) = 2 \times |\Pr(b' = b) - 1/2| \geq \epsilon/l$ and A'' runs in time $O(\text{time}(t))$. \square

Lemma 2. Ψ_{ID} is a secure identity-based encryption scheme if Π_{CL} is secure against chosen-ciphertext attacks.

Proof. (Sketch) Let A be an attacker who can break Ψ_{ID} . Suppose that A has advantage ϵ and runs in time t . We show how to construct from A a Type I adversary A' against Π_{CL} . At the beginning, A' is given by a Π_{CL} challenger a parameter list $params_{cl}$ and six oracles: partial private key exposure oracle $\text{CL_Exp_Partial}_{params_{cl}}^{CLSK^*}(\cdot)$, private key exposure oracle $\text{CL_Exp_Pri}_{params_{cl}}^{CLSK^*}(\cdot)$, public key broadcast oracle $\text{CL_Bro_Pub}_{params_{cl}}(\cdot)$, public key replacement oracle $\text{CL_Rep_Pub}_{params_{cl}}(\cdot, \cdot)$, decryption oracle $\text{CL_Dec}_{params_{cl}}^{CLSK^*}(\cdot, \cdot)$, and left-or-right encryption oracle $\text{CL_Enc}_{params_{cl}}(\cdot, LR(\cdot, \cdot, b))$. To run A , A' simulates the $\text{ID_Gen}(1^k)$ by supplying A with $params = (params_{cl}, CLS_{PKG})$ where CLS_{PKG} is chosen by A' by running $\text{CL_Set_Sec_Val}(params_{cl}, PKG)$. A' keeps a list $L = \{(id, CLD_{id}, CLSK_{id}, CLPK_{id})\}$ where CLD_{id} is the output of partial private key exposure oracle $\text{CL_Exp_Partial}_{params_{cl}}^{CLSK^*}(\cdot)$. The list L does not need to be made in advance and is computed according to the A 's queries. A' responds to A 's oracle queries as follows.

- Key exposure oracle $\text{ID_Exp}_{params}^{IDSK^*}(\cdot)$ queries: Suppose that the request is on a user identity id .
 1. When the list L contains $(id, CLD_{id}, CLSK_{id}, CLPK_{id})$, A' checks to determine whether $CLSK_{id} \neq \perp$ or not. If $CLSK_{id} \neq \perp$, A' returns $IDSK_{id} = CLSK_{id}$ to A . Otherwise, A' sends id to the Π_{CL} partial private key exposure oracle $\text{CL_Exp_Partial}_{params_{cl}}^{CLSK^*}(\cdot)$ and obtains CLD_{id} . A' computes $CLSK_{id} = \text{CL_Set_Pri_Key}(params_{cl}, CLD_{id}, CLS_{PKG})$ and saves $(CLD_{id}, CLSK_{id})$ in the list L . A' returns $IDSK_{id} = CLSK_{id}$.
 2. When the list L does not contain $(id, CLD_{id}, CLSK_{id}, CLPK_{id})$, A' computes $CLPK_{id} = \text{CL_Set_Pub_Key}(params_{cl}, id, CLS_{PKG})$ and

- sends $(id, CLPK_{id})$ to the Π_{CL} public key replacement oracle $CL_Rep_Pub_{params_{cl}}(\cdot, \cdot)$. A' sends id to the Π_{CL} partial private key exposure oracle $CL_Exp_Partial_{params_{cl}}^{CLSK^*}(\cdot)$ and obtains CLD_{id} . A' computes $CLSK_{id} = CL_Set_Pri_Key(params_{cl}, CLD_{id}, CLS_{PKG})$ and adds $(id, CLD_{id}, CLSK_{id}, CLPK_{id})$ to the list L . A' returns $IDSK_{id} = CLSK_{id}$.
- Decryption oracle $ID_Dec_{params}^{IDSK^*}(\cdot, \cdot)$ queries: Suppose that A asks with an input (id, C) .
 1. When the list L contains $(id, CLD_{id}, CLSK_{id}, CLPK_{id})$, A' checks to determine $CLSK_{id} \neq \perp$ or not. If $CLSK_{id} \neq \perp$, A' returns $M = CL_Dec_{params_{cl}}^{CLSK_{id}}(C)$. Otherwise, A' sends (id, C) to the Π_{CL} decryption oracle $CL_Dec_{params_{cl}}^{CLSK^*}(\cdot, \cdot)$ with an optional input CLS_{PKG} ⁱⁱⁱ. Let M be the output of the decryption oracle. A' returns M to A .
 2. When the list L does not contain $(id, CLD_{id}, CLSK_{id}, CLPK_{id})$, A' computes $CLPK_{id} = CL_Set_Pub_Key(params_{cl}, id, CLS_{PKG})$ and sends $(id, CLPK_{id})$ to the Π_{CL} public key replacement oracle $CL_Rep_Pub_{params_{cl}}(\cdot, \cdot)$. A' sets $CLD_{id} = CLSK_{id} = \perp$ and adds $(id, CLD_{id}, CLSK_{id}, CLPK_{id})$ to the list L . A' sends (id, C) to the Π_{CL} decryption oracle $CL_Dec_{params_{cl}}^{CLSK^*}(\cdot, \cdot)$ with an optional input CLS_{PKG} and let M be the output of the decryption oracle. A' returns M to A .
 - Left-or-right encryption oracle $ID_Enc_{params}(\cdot, LR(\cdot, \cdot, b))$ queries: Suppose that A asks with an input (id, M_0, M_1) .
 1. When the list L contains $(id, CLD_{id}, CLSK_{id}, CLPK_{id})$, A' sends (id, M_0, M_1) to the Π_{CL} left-or-right encryption oracle $CL_Enc_{params_{cl}}(\cdot, LR(\cdot, \cdot, b))$ and gets C_b from the Π_{CL} left-or-right encryption oracle $CL_Enc_{params_{cl}}(\cdot, LR(\cdot, \cdot, b))$. A' return C_b to A .
 2. When the list L does not contain $(id, CLD_{id}, CLSK_{id}, CLPK_{id})$, A' computes $CLPK_{id} = CL_Set_Pub_Key(params_{cl}, id, CLS_{PKG})$ and sends $(id, CLPK_{id})$ to the Π_{CL} public key replacement oracle $CL_Rep_Pub_{params_{cl}}(\cdot, \cdot)$. A' sets $CLD_{id} = CLSK_{id} = \perp$ and adds $(id, CLD_{id}, CLSK_{id}, CLPK_{id})$ to the list L . A' sends (id, M_0, M_1) to the Π_{CL} left-or-right encryption oracle $CL_Enc_{params_{cl}}(\cdot, LR(\cdot, \cdot, b))$ and gets C_b from the Π_{CL} left-or-right encryption oracle $CL_Enc_{params_{cl}}(\cdot, LR(\cdot, \cdot, b))$. A' return C_b to A .

Let id_{ch} be the user identity of the query to the left-or-right encryption oracle by Ψ_{ID} attacker A . Before the left-or-right encryption query, A' always sends $(id_{ch}, CLPK_{id_{ch}})$ to the Π_{CL} public key replacement oracle $CL_Rep_Pub_{params_{cl}}(\cdot, \cdot)$ in the simulation process, where $CLPK_{id_{ch}} = CL_Set_Pub_Key(params_{cl}, id_{ch}, CLS_{PKG})$. However, the key exposure oracle query with input id_{ch} never happens. Therefore, A' does not break the adversarial constraints in certificate-less encryption. When the Ψ_{ID} attacker A outputs b' , A' outputs the same b' to the Π_{CL} challenger. Since the A 's view is identical to its view in the real attack, $\text{Adv}_{A', \Pi_{CL}}(k) = 2 \times |\Pr(b' = b) - 1/2| \geq \epsilon$ and A' runs in time $O(\text{time}(t))$. \square

ⁱⁱⁱ The behavior of the Π_{CL} decryption oracle with the optional input is described in Appendix A.

Pre-production Methods of a Response to Certificates with the Common Status –Design and Theoretical Evaluation–

Satoshi Koga¹, Jae-Cheol Ryou², and Kouichi Sakurai³

¹ Graduate School of Information Science and Electrical Engineering,
Kyushu University

6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan
`satoshi@itslab.csce.kyushu-u.ac.jp`

² Division of Electrical and Computer Engineering,
Chungnam National University

220 Gung-dong, Yuseong-gu Daejeon, 305-764, Korea
`jcryou@home.cnu.ac.kr`

³ Faculty of Information Science and Electrical Engineering,
Kyushu University

6-10-1 Hakozaki, Higashi-ku, Fukuoka City, 812-8581, Japan
`sakurai@csce.kyushu-u.ac.jp`

Abstract. The Online Certificate Status Protocol provides the up-to-date response to certificate status queries. To reduce the risk of denial of service attacks, the responder can pre-produce responses. However this approach has the disadvantage that computational costs of the responder are inefficient since the responder should pre-produce one response message for each certificate. This paper proposes efficient pre-producing methods, which the responder can pre-produce a response message for each group consisting of certificates with the common status. In our methods, computational costs of the responder are efficient, compared with the previous pre-producing method.

Keywords: Public Key Infrastructure, Certificate Revocation, Online Certificate Status Protocol, Response Pre-production

1 Introduction

1.1 Background and Motivation

In the *Public Key Infrastructure* (PKI), a certificate is used to bind an entity's identity information with the corresponding public key. Nevertheless, certificates are revoked in case of breaking that binding before its expiration date. Thus, the certificate verifier must check not only the expiration date on the certificate but also the revocation information of it.

A certificate revocation system can be implemented in several ways. The most well-known method is to periodically publish a *Certificate Revocation List* (CRL) [3,4], which is a digitally signed list of revoked certificates and usually issued by

the *Certification Authority* (CA). The main advantage of the CRL systems is its simplicity, but several problems are pointed out [1,17]. Especially, the main disadvantage of the CRL systems is its high communication costs between the user and the repository stored on CRLs, because the size of CRL will be quite long if the CA has many clients.

To overcome the shortcomings of the CRL, several revocation methods are suggested as follows. The Delta CRLs [4] are issued more frequently and only include updates to the complete revocation list called Base CRL. CRL Distribution Points was specified in [4]. CRL Distribution Points allow revocation information within a single domain to be divided into the multiple CRLs. So the CRL of each domain can be smaller than the full CRL. The *Certificate Revocation Tree* (CRT) was proposed by Kocher [7]. CRTs are based on Merkle Hash Trees [9], in which the tree itself represents all certificate revocation information. The status of any given certificate can be proved by using the path from the root to the target leaf. Therefore, the communication costs between the user and repository can be lower than those of CRL systems. Naor and Nissim proposed the Authenticated Directory [14], which improves the reduction in communication cost by balancing the hash tree. They introduced using a 2-3 tree, in which every node has two or three children. In [5,6], the binary hash tree is extended to k -ary hash tree in which any node has at most k children. Micali proposed the revocation system using hash chains [10,11], taking into account both user's and CA's efficiency. The advantage of Micali's system is that the communication costs are very efficient, because the user may just obtain 160-bit hash value.

It is necessary to obtain timely information regarding the revocation status of a certificate. The most popular mechanism that provides real-time status of a certificate is the *Online Certificate Status Protocol* (OCSP) [13]. The OCSP provides the up-to-date response to certificate status queries. Since the user just requests to return the status of certificate, the communication costs can be reduced in comparison to the CRL. The certificate status is returned by a trusted entity referred to as an OCSP responder. The OCSP responder creates signed responses by using own private key. As the extended protocol of the OCSP, the validation protocols which the trusted server builds and validates the certification path instead of the user are proposed [8,16]. In mobile environment, the OCSP appears to be a good choice, because the user can retrieve timely certificate's status with a moderate resource usage. However, there are two threats as follows.

1. *Denial-of-Service* (DoS) attack

In the OCSP, the responder should generate a signed response for every user's request. That is, for every simple request sent by the attacker, the responder must generate the digital signature, which is a computationally intensive operation. Consequently, it becomes highly vulnerable to Denial-of-service (DoS) attacks.

2. Replay attack

The attacker can obtain the old response generated by the responder. By sending the old response, the user misunderstands that revoked certificate is valid. This attack is called replay attack.

The countermeasure against those threats is important and the purpose of this paper is to avoid these attacks.

1.2 Related Work

1. Nonce

To prevent the replay attack, the method of using *nonce* is proposed [13]. The nonce is a random sequence and cryptographically binds a request and a response. The user sends a request message including the nonce. Then the responder must sign a response including the same nonce. (The nonce can be replayed unless it is contained in a signed response message.) The user can prove that the response is fresh by checking that nonce.

However, nonce-based OCSP has the drawback that it is vulnerable to DoS attacks [2]. For every request sent by the attacker, the responder should provide the fresh signed response, that is, the responder should generate the digital signatures. So the attacker can generate enough requests to produce the DoS.

2. Response pre-production

To reduce the risk of DoS attacks, OCSP responders may pre-produce signed responses specifying the status of certificates at a specified time [13]. In this case, the responder does not perform a digital signature for every response, so the computational costs of the responder are more efficient than those of nonce-based OCSP responder. The pre-produced response contains the times : *thisUpdate* and *nextUpdate*. *thisUpdate* is the time at which the status being indicated is known to be correct. *nextUpdate* is the time at or before which newer information will be available about the status of certificate. The user can prove that the response is fresh by checking its validity interval through *thisUpdate* and *nextUpdate*.

However, the use of pre-produced responses allows replay attacks in case that the validity of pre-produced response is long [13]. To avoid the replay attacks, the responder needs to generate pre-produced responses within a short period of time. However, this consumes a lot of processing and this fact may lead the responder against to denial of service. To mitigate the computational costs of the responder, the pre-produced responses are updated using hash-chain [12]. In this scheme, the responder can update the pre-produced responses at low cost.

To prevent the replay attack and DoS attack, this paper focuses on the OCSP response pre-production. In the previous pre-production methods, the responder pre-produces a response message for each certificate. The responder must generate a lot of pre-produced responses if the CA issues many certificates. Additionally, the responder must update pre-produced responses at a short period of time to prevent the replay attack. In case of updating pre-produced responses, the responder must perform the digital signature again. Hence, computational costs of responder will increase and this fact may lead the DoS attack.

1.3 Our Proposed Methods

To reduce the computational costs of the responder, this paper takes a different approach from previous methods toward OCSF response pre-production. This paper proposes efficient pre-production methods, which the responder can pre-produce a response message for each group consisting of certificates with the common status. Then we show that the computational costs of the responder are more efficient than those of the traditional method.

In our methods, certificates with the common status are divided into the multiple groups at first stage. That is, the status of group is “valid” or “revoked”. The responder pre-produces a response message, not for each certificate but for each group. In this paper, we suggest two pre-production methods based on the CRL [4] and CRT [7] descriptions. In our first method based on CRL description, the responder describes certificates contained a group in the response message. While, in our second method based on CRT description, the responder describes the range (a, b) in the response message. (a, b) represents that “certificate a and b are revoked, but any certificates larger than a and less than b are valid.” The responder can pre-produce the response message for these groups.

1.4 Our Result

This paper gives the theoretical evaluation of computational costs. In detail, we evaluate the number of signature generations required for the production and update of pre-produced response. Our evaluations show that the number of signature generations is smaller than that of traditional method. Thus, our methods can reduce the risk of DoS attacks as well as the replay attack.

The issues of our methods are the communication costs and privacy problem. In the traditional OCSF, the size of pre-produced response is constant since one certificate serial number is described in the pre-produced response. But, in our first method based on CRL, the size of pre-produced response will become large because the multiple certificate serial numbers are described in it. But, the computational costs are more efficient than those of traditional OCSF, even if the number of elements contained a group is small. Another issue of our methods is the privacy problem. In the traditional OCSF, the responder sends only the information regarding the revocation status of requested certificate. However, the user can obtain the information about the revocation status of other certificates in our methods. This fact will be leading the privacy concerns.

2 Preliminaries

2.1 OCSF Model

In a OCSF, there are three entities, as shown in Fig. 1.

1. Certification Authority (CA)

A Certification Authority (CA) is a trusted third party that has the responsibility of publishing the certificate revocation information. Compromise of

CA's private key will affect the entire system, so the CA is isolated from the Internet to prevent unauthorized accesses.

2. Responder

A responder is a trusted entity that sends the certificate status information to users. In case of generating the response message containing the status of requested certificate, a responder digitally signs it by own private key. This paper assumes that the status of a certificate is either “valid” or “revoked”.

3. User

Users trust the CA and responder. They send the request message of certificate status to the responder.

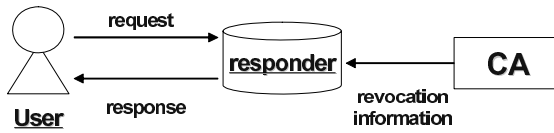


Fig. 1. OCSP Model

2.2 Response Pre-production

To mitigate the risk of DoS attacks, the responders may pre-produce signed responses specifying the status of certificates at a certain time [13]. The pre-produced response message is shown in Fig. 2. The time at which the status was known to be correct shall be reflected in the *thisUpdate* field of the response. The time at or before which newer information will be available is reflected in the *nextUpdate* field. The response pre-production allows the responder to use the same response for multiple requests, without performing a new digital signature for every request. Therefore, the risk of DoS attacks can be reduced.

3 Proposed Pre-production Methods

In the previous methods, the responder pre-produces a response message for each certificate. The responder must generate a lot of pre-produced responses if the CA issues many certificates. Moreover, the responder must update pre-produced responses at a short period of time to prevent the replay attack. Hence, computational costs of responder are inefficient.

This paper takes a different approach from previous methods toward OCSP response pre-production. This paper proposes efficient pre-production methods, which the responder can pre-produce a response message for certificates with the common status. We call certificates with the common status as a “group”. In other words, the status of a group is “valid” or “revoked”. The responder pre-produces signed response message for each group.

Responder ID
Serial Number
Status
Validity (thisUpdate, NextUpdate)
Signature Data

Fig. 2. Pre-produced Response Message

3.1 Decision of the Group

The group consists of certificates with the common status, that is, the status of group is “valid” or “revoked”. Valid certificates may be revoked as time has passed. If some certificates contained in the valid group are revoked, the responder should determine the new group which removed them. Then the responder digitally signs the response message including the new valid group. On the other hand, revoked certificates are determined as a revoked group. As well as the valid group, the responder pre-produces a response message including the new revoked group.

3.2 Our First Method Based on the CRL Description

In the traditional method, the certificate serial number is described in the response message (Fig. 2). In this section, we examine the description of the group.

Our first method uses the same method as the Certificate Revocation List (CRL) [4]. The CRL contains the certificate serial numbers of all revoked certificates. The certificate serial numbers of all certificates contained in the group are described in the response message. When the user receives the response message, she checks whether the serial number of requested certificate is described in the response message or not.

Update processes of groups are described as follows. First, all valid certificates are divided into the valid groups. The responder pre-produces the signed response messages including serial number of certificates contained the valid group. If some certificates contained in the valid group are revoked, the responder creates the new group which removed them (Fig. 3). On the other hand, revoked certificates are determined as one revoked group. As well as the valid group, the responder pre-produces a signed response message for the revoked group.

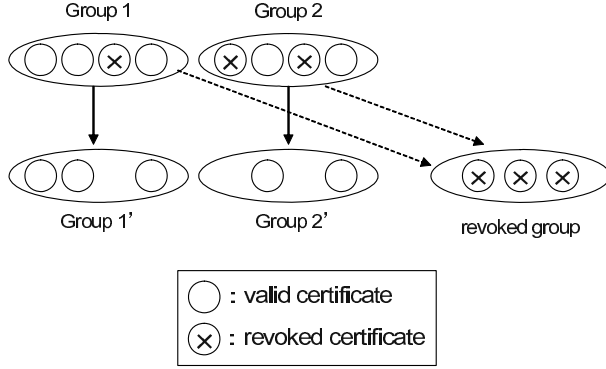


Fig. 3. First method: Update of groups

3.3 Our Second Method Based on CRT Description

Our second method uses the same method as the Certificate Revocation Tree (CRT) proposed by Kocher [7]. In the CRT system, the information from the CRLs is divided into a series of ranges that describe certificate statuses. The range (a, b) is utilized, where a and b denotes the certificate serial number respectively. (a, b) represents that “certificate a and b are revoked, but any certificates larger than a and less than b are valid.” The ranges are used as leaf nodes to build the CRT.

In our method, the responder pre-produces signed response containing the range (a, b) . If the user requests the status of certificate c ($a \leq c \leq b$), she can check the status of requested certificate.

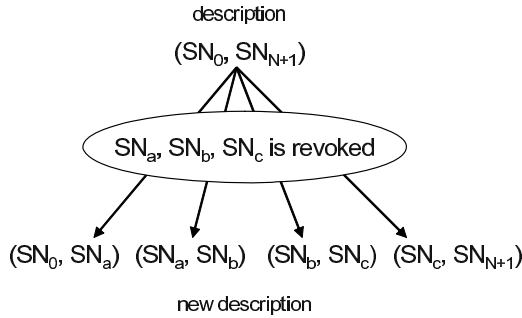


Fig. 4. Second method: Update of groups

This method allows the constant size of response message, but the total number of groups will increase as additional certificates are revoked. For exam-

ple, we assume that the certificates with the serial number SN_1, SN_2, \dots, SN_N are valid. The responder pre-produces a signed response message containing the range (SN_0, SN_{N+1}) . Imagine a simple case where the CA has revoked certificates with serial number SN_a, SN_b, SN_c ($SN_0 < SN_a < SN_b < SN_c < SN_N$), as shown in Fig. 4. In that case, the responder generates the new description $(SN_0, SN_a), (SN_a, SN_b), (SN_b, SN_c), (SN_c, SN_{N+1})$, that is, the number of group is four. Then the responder pre-produces four signed response messages for each group.

4 Theoretical Analysis

The CA publishes the revocation information at some intervals. The responder should update the pre-produced responses by using revocation information. This section analyzes the computational costs of the responder. In detail, we show the total number of signature generation of the responder at some point.

4.1 Notations

- N is the total number of certificates at first stage.
- T is the lifetime of a certificate.
- Δ is the interval at which the CA publishes the revocation information.
- l is the total number of receiving the revocation information during T . That is, $l = T/\Delta$
- p is the revocation rate per Δ .
- k_i is the total number of groups at $i\Delta$.
- C_i is the total number of responder's signing operation at $i\Delta$.

(Assumptions)

1. To prevent the replay attack, the update interval is the same as the lifetime of pre-produced response. So the responder must update each pre-produced response at Δ interval.
2. The CA issues certificates with the serial number $1, 2, \dots, N$. This paper considers that each certificate is valid and has the same lifetime at first stage.

Fig. 5 shows the evaluation model. This paper analyzes C_i as to the traditional method, our first method based on the CRL, and our second method based on the CRT.

4.2 Traditional Method

First, the responder pre-produces the valid response for each certificate because all certificates are valid. That is, C_0 is N .

At Δ , the number of revoked certificates is pN and the responder pre-produces the new responses. If some certificate is revoked, the status of them

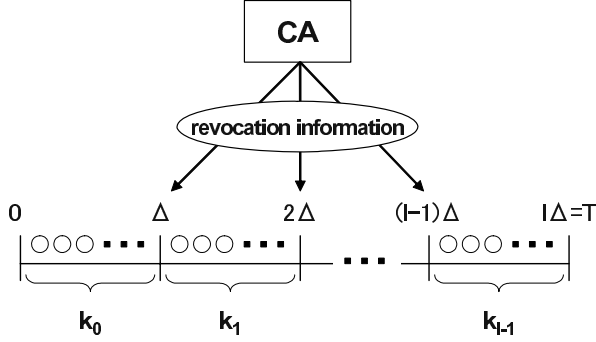


Fig. 5. Evaluation Model

has never been changed within the expiration date of it. So the lifetime of the pre-produced response for each revoked certificate is set as T . In other words, the responder doesn't have to update the pre-produced responses for revoked certificates. While the rest of certificates remain valid, so the responder must update pre-produced response messages for valid certificates. Thus, the responder pre-produces the response messages for revoked certificates and valid certificates. Then C_1 is N as well as C_0 .

Since the total number of valid certificates is $N(1-p)^{i-1}$ at $(i-1)\Delta$, the number of revoked certificates is $pN(1-p)^{i-1}$ from $(i-1)\Delta$ to Δ . The responder must pre-produce the response messages for revoked certificates and valid certificates. Hence, we have the number of signing C_i as follows:

$$C_0 = N, \quad C_i = N(1-p)^{i-1}(1 \leq i \leq l) \quad (1)$$

4.3 Our First Method

At first stage, N certificates are divided into k_0 groups equally. The responder pre-produces the response message for each group, so C_0 is k_0 .

At Δ , the number of revoked certificates is pN . But the pN revoked certificates are determined as one group, so the responder just pre-produces one response message for revoked certificates, as mentioned in Section 3.3. As to the rest of certificates (that is valid certificates), the responder creates the new group which removed revoked certificates. Then the responder pre-produces k_0 response messages for each group. Thus, C_1 is $k_0 + 1$.

Unless all certificates in a group are revoked, the number of group k_i is k_0 at $i\Delta$. Hence, we have the maximum number of signing C_i as follows:

$$C_i = k_0 + 1 \quad (2)$$

4.4 Our Second Method

At first stage, the responder just pre-produces one response message containing the range $(0, N + 1)$ because the certificates with serial number $1, 2, \dots, N$ are valid.

At Δ , pN certificates are revoked and the responder must generate the new group. Unless the certificates with the consecutive serial number are revoked, the number of groups k_1 is $1 + pN$. That is, the responder pre-produces the response message for each group and C_1 is $1 + pN$.

At $(i-1)\Delta$, the number of valid certificate is $N(1-p)^{i-1}$. Then the number of revoked certificates from $(i-1)\Delta$ to $i\Delta$ is $pN(1-p)^{i-1}$. The number of revoked certificate represents the maximum increasing number of the group. Hence, we have the maximum number of signing C_i as follows:

$$C_i = 1 + \sum_{j=1}^i pN(1-p)^{j-1} = 1 + N\{1 - (1-p)^i\} \quad (3)$$

5 Evaluation

In this section, we evaluate the computational costs of our methods by using the equations (1), (2), and (3).

5.1 Parameters

1. $N = 1000000$.
2. $l = 365 \times 24 \times 60$.

We assume that the CA publishes the revocation information for one minutes interval.

3. $p = 1 - \sqrt[l]{1 - P}$

P denotes the revocation rate per year. It is said that a certificate revocation rate around 10 percent per year is reasonable [15], so we assume that $P = 0.1$.

5.2 Computational Costs

Fig. 6 shows the comparison of C_i between the traditional method and our first method. In the traditional method, C_i is reduced as the number of revoked certificates increases. On the other hand, C_i is independent of the number of revoked certificates in our first method. If the number of certificates contained in the group is more than or equal to two (that is, k_0 is less than or equal to $N/2$), our first method allows the low computational costs of responder, compared with those of the traditional method.

Fig. 7 shows the comparison of C_i between the traditional method and our second method. In our second method, C_i is increases as the number of revoked certificate increases. If the revocation rate per one year is more than 0.5, the

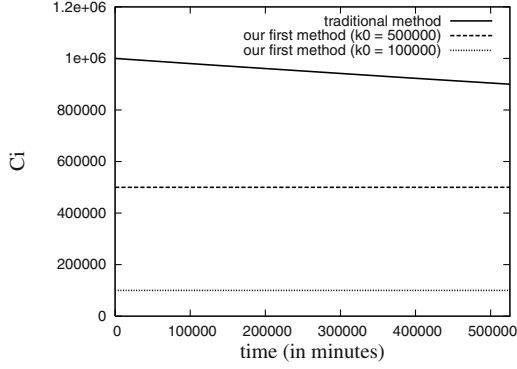


Fig. 6. Computational Costs: Comparison between the traditional method and our first method

computational costs is higher than those of the traditional method (Fig. 8). However, $P > 0.1$ is not realistic value. Hence, the computational costs are more efficient in comparison to those of the traditional method.

5.3 Communication Costs

In this section, we discuss the communication costs. In detail, we examine the size of pre-produced response.

1. Traditional method

In the traditional method, the responder just describes one certificate serial number in the response message and the size of response message is a constant.

2. Our first method

The responder should describe the multiple certificate serial numbers in the response message. The size of response message is dependent on k_0 . If the k_0 is small, the size of response message may become large.

3. Our second method

The responder just describes a range (a, b) in the response message. Therefore, the size of response message is constant.

5.4 Privacy

Another issue of our methods is the privacy problem. In the traditional OCSP, the responder sends only the information regarding the revocation status of requested certificate. However, the user can obtain the information about the revocation status of other certificates. This fact will be leading the privacy concerns as well as the CRL system.

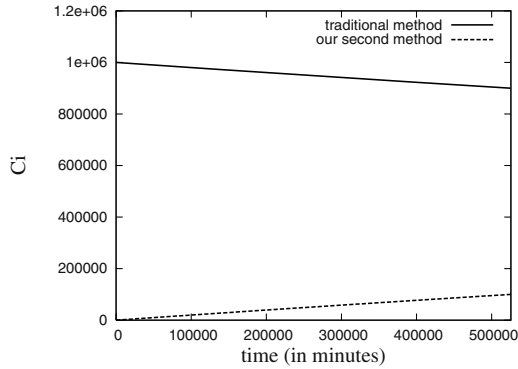


Fig. 7. Computational Costs: Comparison between the traditional method and our second method ($P=0.1$)

6 Conclusion

To avoid the DoS and replay attacks, this paper focuses the response pre-production in OSCP. The previous response pre-productions have the drawbacks that the computational costs of a responder are inefficient if the number of certificates is large. This paper proposes efficient pre-producing methods, which the responder can pre-produce a response message for each group consisting of certificates with the same status. In our methods, computational costs of the responder are low, compared with the previous pre-producing method.

Acknowledgements

The second author of this research was supported by University IT Research Center Project of Korea MIC (Ministry of Information and Communication).

References

1. A. Arnes, M. Just, S. J. Knapskog, S. Lloyd, and H. Meijer, *Selecting Revocation Solutions for PKI*, 5th Nordic Workshop on Secure IT Systems (NORDSEC 2000), 2000. <http://www.pvv.ntnu.no/~andream/certrev/>
2. CoreStreet Ltd., *Nonce Sense - Freshness and Security in OSCP Responses*-, CoreStreet Ltd. White Paper, 2003. <http://www.corestreet.com/whitepapers/nonce-sense.pdf>
3. R. Housley, W. Polk, W. Ford, and D. Solo, *Certificate and Certificate Revocation List (CRL) Profile*, IETF RFC3280, 2002. <http://www.ietf.org/rfc/rfc3280.txt>
4. ITU/ISO Recommendation. X.509 Information Technology Open Systems Interconnection - The Directory: Authentication Frameworks, 1997.

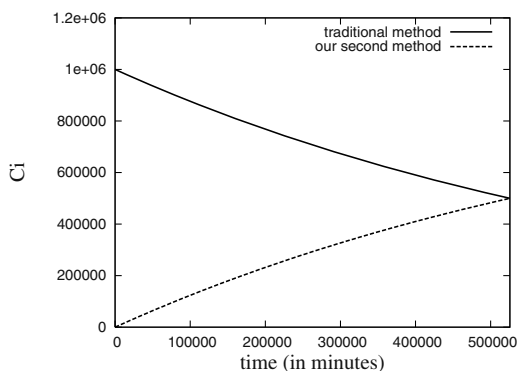


Fig. 8. Computational Costs: Comparison between the traditional method and our second method ($P=0.5$)

5. H. Kikuchi, K. Abe, and S. Nakanishi, *Performance Evaluation of Certificate Revocation Using k -Valued Hash Tree*, 2nd International Workshop on Information Security (ISW '99), LNCS 1729, pp.103-117, Springer-Verlag, 1999.
6. H. Kikuchi, K. Abe, and S. Nakanishi, *Certificate Revocation Protocol Using k -Ary Hash Tree*, IEICE TRANS. COMMUN., Vol. E84-B, No.8, 2001.
7. P. C. Kocher, *On Certificate Revocation and Validation*, Financial Cryptography (FC '98), LNCS 1465, pp.172-177, Springer-Verlag, 1998.
8. A. Malpani, R. Housley, and T. Freeman, *Simple Certificate Validation Protocol*, IETF Internet-Draft, 2003.
<http://www.ietf.org/internet-drafts/draft-ietf-pkix-scvp-13.txt>
9. R. C. Merkle, *A Certified Digital Signature*, Advances in Cryptology - CRYPTO '89, LNCS 435, pp.218-238, Springer-Verlag, 1990.
10. S. Micali, *Efficient Certificate Revocation*, Technical Memo MIT/LCS/TM-542b, Massachusetts Institute of Technology, 1996.
11. S. Micali, *NOVOMODO; Scalable Certificate Validation And Simplified PKI Management*, 1st Annual PKI Research Workshop, pp.15-25, 2002.
<http://www.cs.dartmouth.edu/~pki02/>
12. J. L. Munoz, J. Forne, O. Esparza, I. Bernable, and M. Soriano, *Using OCSP to Secure Certificate-Using Transactions in M-commerce*, Applied Cryptography and Network Security (ACNS 2003), LNCS 2846, pp.280-292, Springer-Verlag, 2003.
13. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol-OCSP*, IETF RFC2560, 1999. <http://www.ietf.org/rfc/rfc2560.txt>
14. M. Naor, and K. Nissim, *Certificate Revocation and Certificate Update*, 7th USENIX Security Symposium, pp.217-228, 1998.
<http://www.usenix.org/publications/library/proceedings/usenix98/>
15. A. Nash, W. Duane, C. Joseph, and D. Brink, *PKI - Implementing and Managing E-Security*, Osborne Media Group, 2001.
16. D. Pinkas, and R. Housley, *Delegated Path Validation and Delegated Path Discovery Protocol Requirements*, RFC3379, 2002. <http://www.ietf.org/rfc/rfc3379.txt>
17. R. L. Rivest, *Can We Eliminate Certificate Revocation Lists?*, Financial Cryptography (FC '98), LNCS 1465, pp.178-183, Springer-Verlag, 1998.

Filling the Gap between Requirements Engineering and Public Key/Trust Management Infrastructures^{*}

Paolo Giorgini¹, Fabio Massacci¹, John Mylopoulos^{1,2}, and Nicola Zannone¹

¹ Department of Information and Communication Technology
University of Trento - Italy

{massacci, giorgini, zannone}@dit.unitn.it

² Department of Computer Science
University of Toronto - Canada
jm@cs.toronto.edu

Abstract. The last years have seen a major interest in designing and deploying trust management and public key infrastructures. Yet, it is still far from clear how one can pass from the organization and system requirements to the actual credentials and attribution of permissions in the PKI infrastructure.

Our goal in this paper is filling this gap. We propose a formal framework for modeling and analyzing security and trust requirements, that extends the Tropos methodology for early requirements modeling. The key intuition that underlies our work is the identification of distinct roles for actors that manipulate resources, accomplish goals or execute tasks, and actors that own or permit usage of resources or goals. The paper also presents a simple case study and a PKI/trust management implementation.

Keywords: Security Engineering, Modelling and Architecture, Verification, Privilege Management, PKI and eHealth applications, PKI Requirements Analysis, Trust.

1 Introduction

Trust Management and PKIs are hot topics in security research [5, 4, 7, 10, 14, 18]. There are a number of sophisticated policy languages (e.g., [8]), algorithms, and system for managing security credentials. The trust-management approach has a number of advantages over other mechanisms for specifying and controlling authorization, especially when security policy is distributed over a network or is otherwise decentralized.

Solutions based on public-key cryptography and credential have been shown to be well suited in satisfying the security requirements of distributed systems and becoming the foundation for those applications that require security authentication. The reason is that it is impractical and unrealistic to expect that each user in a large scale system has a previously established relationship with all other users.

^{*} This work has been partially funded by the IST programme of the EU Commission, FET under the IST-2001-37004 WASP project and by the FIRB programme of MIUR under the RBNE0195K5 ASTRO Project. We would like to thank the anonymous reviewers for useful comments.

However, if we look at the connection between these credential-based systems and the requirements of the entire IT system we find a large gap. There are no methodologies for linking security policy to the mainstream requirements analysis process. This might be an instance of the general problem of security engineering. The usual approach towards the inclusion of security within a system is to identify security requirements after system design. This is a critical problem [2], mainly because security mechanisms have to be fitted into a pre-existing design which may not be able to accommodate them [25]. Late analysis of security requirements can also generate conflicts between security needs and functional requirements of the system. Even with the growing interest in secure engineering, current methodologies for information system development do not meet the needs for resolving the security related IS problem [26].

In the literature there are proposals improving on secure engineering (see [13, 15, 21, 23]) or architectures for trust management (see [5, 4, 14, 18, 22]), but nobody has proposed a methodology that considers together both these approaches. Our goal is to introduce a trust management system into the requirements engineering framework. Essentially, we would like to avoid designing an entire IT system and then retrofitting a PKI on its top, when it is already too late to make it fit snugly.

In this paper we introduce a process that integrates trust, security and system engineering, using the same concepts and notations used for requirements specification. To devise the PKI/trust management structure, we propose to proceed in three steps. First, we build a functional requirements model where we show functional dependencies among actors, then we give a trust requirements model, where we study whether trust relationships among actors correspond to security requirements. Finally, we built a PKI/trust management implementation where the designer can define credentials and delegations certificates confronting them with the relationships captured in the other models and checking whether an actor that offers a service is authorized to have it.

The paper is organized as follows. Next (§2) we provide a brief description of Tropos methodology and describe the basic concepts and diagrams that we use for modeling security. We introduce a simple Health Care Information System (§3) that will be used as case study throughout the paper. Then we present in the formalization of the security notions (§4) and define axioms and properties of our framework (§5). Next (§6) we introduce negative authorizations. Then (§7) we define the trust implementation of our framework into the RT framework. Finally, we conclude the paper with some directions for future work (§8).

2 Security-Aware Tropos

The first step towards security engineering is to model the entire organization and procedures. Security failures often are organizational or procedural failures [2]. Thus, we have chosen a requirement framework that allows for the modeling of the entire organization: Tropos [6].

Tropos is an agent-oriented software development methodology, tailored to describe both the organization and the system. One of the main feature of Tropos is the crucial role given to the early requirements analysis that precedes the prescriptive requirements specification. The main advantage of this is that, by doing an earlier analysis, one can

capture not only the *what* or the *how*, but also the *why* a piece of software is developed. This, in turn, supports a more refined analysis of the system dependencies and, in particular, for a much better and uniform treatment, not only of the system's functional requirements, but also of the non-functional requirements (the latter being usually very hard to deal with).

Tropos uses the concepts of actor, goal, soft goal, task, resource and social dependency for defining the obligations of actors (dependees) to other actors (dependers). Actors have strategic goals and intentions within the system or the organization and represent (social) agents (organizational, human or software), roles or positions (that represent a set of roles). A goal represents the strategic interests of an actor. A task specifies a particular course of action that produces a desired effect, and can be executed in order to satisfy a goal. A resource represents a physical or an informational entity. Finally, a dependency between two actors indicates that one actor depends on another to accomplish a goal, execute a task, or deliver a resource. In the rest of the paper, we say service for goal, task, or resource. For example, Yu et al. [20] have used the Tropos framework to model strategic goal concerning privacy and security of agents and have used formal tools for some reachability and goal analysis.

Although Tropos is based on the agent paradigm, it can be also combined with non-agent (e.g., object-oriented) software development paradigms. For example, one may want to use Tropos for early development phases and then use UML-based approaches (e.g., the Rational Unified Process). Tropos can be also combined with more formal approaches, like for instance [3], allowing so for the description of the dynamic aspects and the verification of requirements specification. There is a considerable amount of work in this direction within the Formal Tropos project [11].

After the Tropos formalization we are still far behind capturing security requirements, because Tropos has been designed with cooperative information systems in mind. Thus a dependency between two actors means that the dependee will take responsibility for fulfilling the functional goal of a depender, but we have no way to specify or check that it is actually authorized to do so. Thus, we identify four relationships:

- *trust*, among two agents and a service,
- *delegation*, among two agents and a service,
- *ownership*, between an agent and a service, and
- *offer*, between an agent and a service.

Note the difference between owning a service and offering a service. For example, a patient is the legitimate owner of his personal data. However, the data may be stored on a medical information system that offers access to the data. This distinction explains clearly why IS managers need the consent of the patient for data processing. Also note the difference between trust and delegation. Delegation marks a formal passage in the requirements modeling. In contrast, trust marks simply a social relationship that is not formalized by a “contract” (such as digital credential).

Moreover, we do not assume that a delegation implies a trust. Using this extension of the modeling framework, we can now refine the process:

1. define functional dependencies of services among agents;
2. design a trust model among the actors of the systems;
3. identify who owns services and who is able to fulfill them.

3 An Illustrative Case Study

We present a simple case of health care IS to illustrate our approach. This example derives from EU privacy legislation: a citizen's personal data is processed by an information system (which offer a data access service) but it is owned by the citizen himself whose consent is necessary for the service to be delivered to 3rd parties ³.

We consider the following actors:

- *Patient*, that depends on the hospital for receiving appropriate health care. Further, patients will refuse to share their data if they do not trust the system or do not have sufficient control over the use of their data;
- *Hospital*, that provides medical treatment and depends on the patients for having their personal information.
- *Clinician*, physician of the hospital that provides medical health advice and, whenever needed, provide accurate medical treatment;
- *Health Care Authority* (HCA) that control and guarantee the fair resources allocation and a good quality of the delivered services.
- *Medical Information System* (MIS), that, according the current privacy legislation, can share the patients medical data if and only if consent is obtained. The *MIS* manages patients information, including information about the medical treatments they have received.

In order to provide rapid and accurate medical treatments, clinicians need a fast access to their patient' medical data. Similarly, HCA needs a fast and reliable access to the data in order to allocate effectively the available resources, and guaranteeing then that each patient can receive a good quality of medical care. Furthermore, HCA wants to be sure that the system cannot be defrauded in any way and that clinicians and patients behave within the limits of their roles. To the other hand, the obvious right of the patient to restrict access on his/her medical data and moreover, to be able to use some safeguards on the privacy of these data, should be taken into serious consideration. The patient's consent must be requested, and he must be notified when its data is shared.

Figure 1 shows the functional requirement model. In the functional requirement model we represent a (Tropos) dependence as an edge labelled by **D**. For every actor we show the goals that they have to aim (**A**) and the services they can offer (**S**). Then we built the trust requirement model. The basic idea is that the owner of an object has full authority concerning access and disposition of his object, and he can also delegate it to other actors. We represent this relationship as an edge labelled by **O**. Further, we want separate the concept of authority from the concept of permission. This allows to use the notion of authority as a prerequisite for creating permissions. By expressing constraints on future delegations one defines the scope of future management structure in an organization. To this end, we introduce the notion of trust and delegation. The basic meaning of trust is to determine whether a actor is authorized to have the object. Thus, we use trust (**T**) to model the basic trust relationship between agents. In the trust management implementation we use delegation to model the actual transfer of rights in some form (e.g. a digital certificate, a signed paper, etc.etc.). We consider two kind

³ Of course, this is not true for most countries in the world.

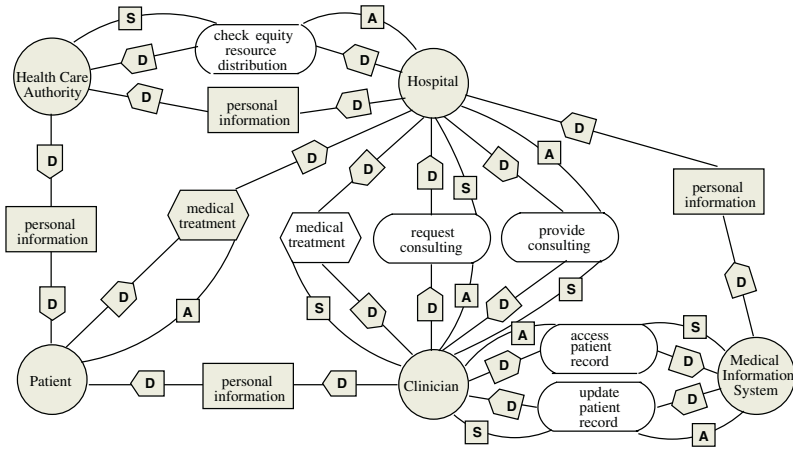


Fig. 1. Health Care System functional requirement model

of delegation: delegation for use, or permission (**P**), and delegation for grant (**G**). We believe that both types of delegation should be offered by an information system since a choice would then result in greater flexibility. Essentially we assume that if an actor is a delegatee for a certain service it must have appropriate permission from the owner of the service. The *Patient* aims to get medical treatments. *Hospital* aims to check equity resource distribution. To the other hand, *Clinician* can offer medical treatments and *HCA* can check equity resource distribution. So, *Patients* depend on the *Hospital* for receiving medical treatments, and in turn, *Hospital* depends on *Clinician* for providing such treatments. *Clinician* depends on the *Patients* for their personal information and on the *Hospital* for specific professional consultancies. The *Hospital* depends on other *Clinicians* for providing professional consultancies and on *HCA* for checking equity resource distribution and for patient personal information. *HCA* depends on *Patient* for personal information. Finally, we also consider the dependencies between *Clinician* and *MIS* to access patient record and to update patient record.

Figure 2 shows the trust requirement model. The *Patient* owns his personal information and *Clinician* owns medical treatments. *Patient* trusts *HCA* and *Clinician* for his personal information, and *HCA* trusts *Hospital* for it. Further, *Hospital* trusts *HCA* for checking equity resource distribution. *Clinician* trusts *Hospital* for medical treatment and for requesting specific professional consulting, and *Hospital* trusts *Clinician* for providing such consulting. Notice on top of Fig. 2 that there is a trust relationship between two actors (*HCA* and *Hospital*) on a resource that is owned by neither of them.

Figure 3 shows the trust management implementation. *Clinician* delegates for grant medical treatments to *Hospital* and *Hospital* delegates for use the goal to check equity resource distribution to *HCA*. *Clinician* and *HCA* need patient personal information to fulfill their service. Thus, *Patient* delegates them his personal information. Further, *HCA* delegates for grant these data to *Hospital*, and in turn, *Hospital* delegates for grant them to *MIS*.

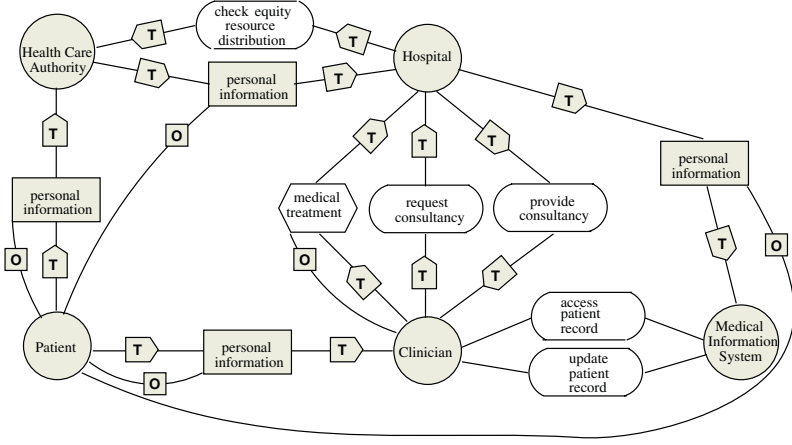


Fig. 2. Health Care System trust requirement model

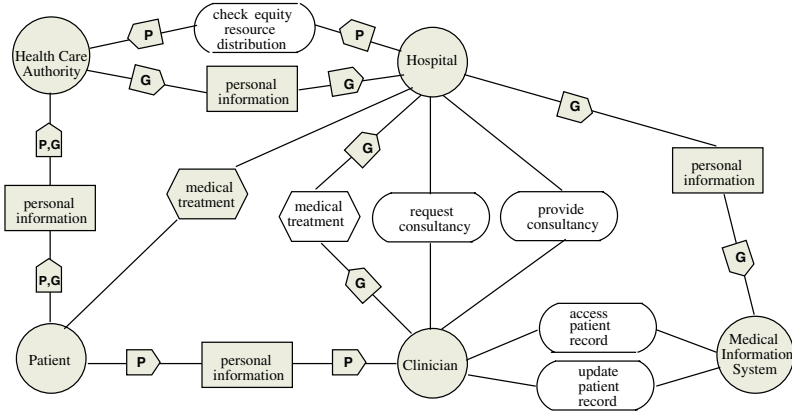


Fig. 3. Health Care System trust management implementation

4 Formalization

To build the formal semantics of the requirements model, we use a form of delegation logics to model security requirements. Particularly, we follow Li et al. [16, 17] that provides a logical framework for representing security policies and credentials for authorization in large-scale, open, distributed systems.

We start by presenting the set of predicates for the functional requirement model. When an actor has the capabilities to fulfill a service, he offers it. The intuition is that $\text{offers}(a, s)$ holds if instance a offers the corresponding instance of s . We assume that a can offer the service if he has it. The predicate $\text{aims}(a, s)$ holds if actor a has the objective of reaching to fulfill the goal s . The predicate $\text{depends}(a, b, s1, s2)$ holds if actor a depends from actor b for service $s1$ for the fulfillment of service $s2$.

Functional Requirement Model
offers(Actor : a , Service : s)
aims(Actor : a , Service : s)
depends(Actor : a , Actor : b , Service : $s1$, Service : $s2$)
Trust Requirement Model
owns(Actor : a , Service : s)
trust(Actor : a , Actor : b , Service : $s1$, Service : $s2$, $\mathcal{N}^+ \cup \{*\}$: n)
Trust Management Implementation
has(Actor : a , Service : s)
fulfills(Actor : a , Service : s)
delGrant(id : idC , Actor : a , Actor : b , Service : $s1$, Service : $s2$, $\mathcal{N}^+ \cup \{*\}$: n)
permission(id : idC , Actor : a , Actor : b , Service : $s1$, Service : $s2$)

Table 1. Predicates

Next we have predicates for the trust requirement model. The predicate $owns(a, s)$ holds if the instance a owns the service s . The owner of a service has full authority concerning access and usage of his services, and he can also delegate this authority to other actors. The predicate $trust(a, b, s1, s2, n)$ holds if actor a trusts actor b for service $s1$ to fulfill service $s2$; n is called *trust depth*. As suggest by Li et al. [16] for their delegation logics, trust has depth, which is either a positive integer or “*” for unbounded depth.

Finally, we use the following predicates to model the trust management implementation. The basic idea of has is that who has a service, has authority concerning access and disposition of the service, and he can also delegate this authority to other actors provided the owner of the service agree. The predicate $fulfills(a, s)$ holds if actor a fulfills the service s . Every trust management framework is based on credentials and delegation. We distinguish two predicates: $delGrant$ and $permission$. The intuition is that $delGrant(idC, a, b, s1, s2, n)$ holds if actor a delegates the permission to grant the service $s1$ to fulfill the goal $s2$ to actor b . The intuition is that $permission(idC, a, b, s1, s2)$ holds if actor a delegates the permission to use the service $s1$ to reach the goal $s2$ to actor b . The actor a is called the *delegator*; the actor b is called the *delegatee*; idC is the certificate identify; n is called the *delegation depth*. A delegation has depth as for trust. One way to view depth is the number of re-delegation steps that are allowed; depth 1 means that no re-delegation is allowed, depth N means that $N - 1$ further step are allowed, and depth “*” means that unbounded re-delegation is allowed. We abbreviate delegation and permission chain as follows

$$\begin{aligned}
delGChain(A, B, S1, S2) &\equiv \left\{ \begin{array}{l} \exists k \text{ s.t. } \exists a_1 \dots a_k \exists n_1 \dots n_{k-1} \forall i \in [1 \dots k-1] \\ delGrant(id_i, a_i, a_{i+1}, S1, S2, n_i) \wedge a_1 = A \wedge a_k = B \end{array} \right. \\
permissionChain(A, C, S1, S2) &\equiv \left\{ \begin{array}{l} (permission(idC, A, C, S1, S2)) \vee \\ (\exists B \ delGChain(A, B, S1, S2) \wedge \\ permission(idC, B, C, S1, S2)) \end{array} \right.
\end{aligned}$$

Functional Requirement Model
Ax1: $\text{aims}(B, S1) :- \text{depends}(A, B, S1, S2)$
Trust Requirement Model
Ax2: $\text{trust}(A, B, S1, S2, N - 1) :- \text{trust}(A, B, S1, S2, N) \wedge N > 2$
Ax3: $\text{trust}(A, C, S1, S2, P) :- \text{trust}(A, B, S1, S2, N) \wedge \text{trust}(B, C, S1, S2, M) \wedge P = \min\{N - 1, M\} \wedge N > 2$
Trust Management Implementation
Ax4: $\text{has}(A, S) :- \text{owns}(A, S)$
Ax5: $\text{has}(B, S1) :- \text{delGrant}(ID, A, B, S1, S2, N)$
Ax6: $\text{has}(B, S1) :- \text{permission}(ID, A, B, S1, S2)$
Ax7: $\text{fulfills}(A, S) :- \text{has}(A, S) \wedge \text{offers}(A, S)$
Ax8: $\text{fulfills}(A, S1) :- \text{depends}(A, B, S1, S2) \wedge \text{fulfills}(B, S1)$
Ax9: $\text{fulfills}(A, S) :- \forall S' \sqsubseteq S, \text{fulfills}(A, S')$

Table 2. Axioms

5 Axioms and Properties

In order to illustrate our approach we formalize the case study and check-model it in Datalog [1]. A datalog program is a set of rules of the form $L :- L_1 \wedge \dots \wedge L_n$ where L , called head, is a positive literal and L_1, \dots, L_n are literals and they are called body. Intuitively, if L_1, \dots, L_n are true in the model then L must be true in the model. In Datalog, negation is treated as negation as failure. In other words, if there is no evidence that an atom is true, it is considered to be false, and hence if an atom is not true in some model, then its negation should be considered to be true in that model. In this way, if a subgoal is not fulfilled, also the correspondent main goal is not fulfilled.

The intuitive descriptions of systems are often incomplete, and need to be completed for a correct analysis. To draw the right conclusions from an intuitive model, we need to complete the model using a systematic method. To this end we use *axioms*.

In Table 2 we present the axioms for our framework. Ax1 says that if an actor depends to other actors to fulfill a service the last has as objective the service. Ax2 states that if someone trust with depth N , then he trust with smaller depth. Ax3 completes the trust relationship between actors. As we say before, the owner of a service has full authority concerning access and disposition of it. Thus, Ax4 states that if an actor owns a service, he has it. Ax5 and Ax6 say that the delegatee has the service. Ax7 states that if an actor has a service and offers it, then he fulfills the service. Ax8 says that if an actor depends to another and the second fulfills the service, also the first fulfill the service. Ax9 is for and-decomposition and states that an actor fulfills the main service if he has fulfilled all its subservices. For or-decomposition the main goal is fulfilled if one of its subgoals is fulfilled. Note that $S' \sqsubseteq S$ means that S' is subgoal of S .

Properties are different from axioms: they are design feature that must be checked. If the set of features is not consistent, i.e. they cannot all be simultaneously satisfied, the system is inconsistent, and hence it is not secure. In Table 3 we use the $A \Rightarrow? B$ to mean that one must check that each time A holds it is desirable that B also holds. In Datalog this can be represented as the constraint $:- A, \text{ not } B$.

Pro1:	$\text{aims}(A, S) \Rightarrow? \text{fulfills}(A, S)$
Pro2:	$\text{has}(B, S1) \wedge \text{owns}(A, S1) \wedge A \neq B \Rightarrow? \exists N \text{ trust}(A, B, S1, S2, N)$
Pro3:	$\text{fulfills}(B, S1) \wedge \text{owns}(A, S1) \wedge A \neq B \Rightarrow? \exists N \text{ trust}(A, B, S1, S2, N)$
Pro4:	$\text{has}(B, S1) \wedge \text{owns}(A, S1) \wedge A \neq B \Rightarrow? \begin{cases} \text{delGChain}(A, B, S1, S2) \vee \\ \text{permissionChain}(A, B, S1, S2) \end{cases}$
Pro5:	$\text{fulfills}(B, S1) \wedge \text{owns}(A, S1) \wedge A \neq B \Rightarrow? \text{permissionChain}(A, B, S1, S2)$
Pro6:	$\text{fulfills}(A, S) \Rightarrow? \text{has}(A, S)$
Pro7:	$\text{permission}(ID, A, B, S1, S2) \Rightarrow? \text{has}(A, S1)$
Pro8:	$\text{delGrant}(ID, A, B, S1, S2, N) \Rightarrow? \text{has}(A, S1)$
Pro9:	$\text{permission}(ID, A, B, S1, S2) \Rightarrow? \exists N \text{ trust}(A, B, S1, S2, N)$
Pro10:	$\text{delGrant}(ID, A, B, S1, S2, N) \Rightarrow? \exists M \geq N \text{ trust}(A, B, S1, S2, M)$
Pro11:	$\text{permissionChain}(A, B, S1, S2) \Rightarrow? \exists N \text{ trust}(A, B, S1, S2, N)$
Pro12:	$\text{delGChain}(A, B, S1, S2) \Rightarrow? \exists N \text{ trust}(A, B, S1, S2, N)$
Pro13:	$\text{delGChain}(A, B, S1, S2) \Rightarrow? \begin{cases} \exists M \exists A_1 \dots A_M \exists N_1 \dots N_{M-1} \\ \forall i \in [1 \dots M-1] \\ \text{delGrant}(ID_i, A_i, A_{i+1}, S1, S2, N_i) \wedge \\ A_1 = A \wedge A_M = B \wedge N_i > N_{i+1} \end{cases}$

Table 3. Desirable Properties of a Design

Table 3 shows a number of properties. Pro1 wants to check if an actor fulfills the services that he has as objective. Pro2 and Prop3 state that if an actor has or fulfills a service and it belongs to another actor, the last has to trust first one. Pro4 and Prop5 state that if an actor has or fulfills a service and it belongs to another actor, there is a delegation chain from the first to the second. Pro6, Pro7, and Pro8 state that if an agent fulfills or delegates a service, he should have it. Pro9, Pro10, Pro11, and Pro12 state that an actor who delegates something to other or there is a delegation chain, the delegator has to trust the delegatee. Rights or privileges can be given to trusted agents that are then responsible for agents they may delegate this right to. So the agents will only delegate to agents that they trust. This forms a delegation chain. If any agent along this chain fails to meet the requirements associated with a delegated right, the chain is broken and all agents following the failure are not permitted to perform the action associated with the right. Thus, Prop13 is used to verify whether the delegate chain is valid.

As already proposed in [12], our framework supports automatic verification of security requirements. Particularly, we use the DLV system [9] to check system consistency.

6 Negative Authorizations

In all practical example of policies and requirements for e-health we found the need for negative authorization (for non-functional requirements) and negative goals or goal whose fulfillment obstacles the fulfillment of other goals (for functional requirements). Tropos already accommodates the notion of positive or negative contribution of goals to the fulfillment of other goals. We only need to lift the framework to permission and trust. Notice that having negative authorization in the requirements model does *not* mean that we must use “negative” certificates. Even if some form of negative certificates are often

Trust Management Implementation
delDenial($id : idC, Actor : a, Actor : b, Service : s, \mathcal{N}^+ \cup \{*\} : n$)
prohibition($id : idC, Actor : a, Actor : b, Service : s$)

Table 4. Negative Authorization Predicates

used in real life⁴ we can use negative authorization to help the designer in shaping the perimeter of positive trust, i.e. positive certificates, to avoid incautious delegation certificates that may give more powers than desired.

We use a *closed world* policy. Under this policy, the lack of an authorization is interpreted as a negative authorization. Therefore, whenever a actor tries to access an object, if a positive authorization is not found in the system, the actor is denied the access. This approach has a major problem in the lack of a given authorization for a given actor does not prevent this user from receiving this authorization later on.

Suppose that an actor should not be given access to a service. In situations where authorization administration is decentralized, an actor possessing the right to use the service, can delegate the authorization on that service to the wrong actor. Since many actors may have the right to use a service, it is not always possible to enforce with certainty the constraint that a actor cannot access a particular service. We propose an explicit *negative* authorization as an approach for handling this type of constraint.

An explicit negative authorization express a *denial* for an actor to access a service. In our approach negative authorizations are stronger than positive authorizations. That is, whenever a user has both a positive and a negative authorization on the same object, the user is prevented from accessing the object. Negative authorizations in our model are handled as blocking authorizations. Whenever a user receives a negative authorization, his positive authorizations become blocked. We distinguish two predicates: delDenial and prohibition. The intuition is that delDenial(idC, a, b, s, n) holds if actor a delegates the permission to denial the service s to actor b . The intuition is that prohibition(idC, a, b, s) holds if actor a forbids to use the service s to actor b . Actor a says that service s cannot be assigned to actor b . We assume that if an actor a denial an actor b to have service s there is not a delegation chain from a to b . So, if a is the owner of s then b cannot have s . Otherwise b could have s if there exists a delegation chain from owner of s and b without a . As done for positive authorization, we can define an abbreviation for a denial chain as

$$\text{delDChain}(A, B, S) \equiv \left\{ \begin{array}{l} \exists k \text{ s.t. } \exists a_1 \dots a_k \exists n_1 \dots n_{k-1} \forall i \in [1 \dots k-1] \\ \text{delDenial}(id_i, a_i, a_{i+1}, S, n_i) \wedge a_1 = A \wedge a_k = B \end{array} \right.$$

and an abbreviation for a prohibition chain as

$$\text{prohibitionChain}(A, C, S) \equiv \left\{ \begin{array}{l} (\text{prohibition}(idC, A, C, S)) \vee \\ (\exists B \text{ delDChain}(A, B, S) \wedge \text{prohibition}(idC, B, C, S)) \end{array} \right.$$

⁴ E.g., A certificate issued by the government that you have no pending criminal trials

Trust Management Implementation	
Ax5:	$\text{has}(B, S1) :- \text{delGrant}(ID, A, B, S1, S2, N) \wedge \text{not prohibitionChain}(A, B, S1)$
Ax6:	$\text{has}(B, S1) :- \text{permission}(ID, A, B, S1, S2) \wedge \text{not prohibitionChain}(A, B, S1)$

Table 5. Negative Authorization Axioms

Pro14:	$\text{prohibition}(ID, A, B, S) \wedge \text{owns}(A, S) \Rightarrow ? \text{not has}(B, S)$
Pro15:	$\text{prohibition}(ID, A, B, S) \wedge \text{owns}(A, S) \Rightarrow ? \text{not fulfills}(B, S)$
Pro16:	$\text{delDChain}(A, B, S) \Rightarrow ? \left\{ \begin{array}{l} \exists M \exists A_1 \dots A_M \exists N_1 \dots N_{M-1} \\ \quad \forall i \in [1 \dots M-1] \\ \text{delDenial}(ID_i, A_i, A_{i+1}, S, N_i) \wedge \\ A_1 = A \wedge A_M = B \wedge N_i > N_{i+1} \end{array} \right.$

Table 6. Negative Authorization Desirable Properties

As we say for positive authorization, the intuitive description of systems are often incomplete, and need to be completed for providing correct analysis. To this end we use axioms to complete the model. In Table 5 we show how. Ax5 and Ax6 are modified to account for the possibility of negative authorizations: we have to add in the body of the rules that there is not a prohibition chain from the delegator to the delegatee.

System designer should check that the system is secure. Table 6 presents properties for negative authorization to verify if the model respects security features. Pro14 and Pro15 check that if the owner of a service forbids to use it to another actor, the last one cannot have and fulfill the service. Pro16 is used to verify if a denial chain is valid.

7 Trust Management Implementation

Several trust management systems [5, 4, 16] have been proposed to address authorization in decentralized environments. In this section we present the implementation of our approach using the RT (Role-based Trust-management) framework [17, 18, 19].

The RT framework provides policy language, semantics, deduction engine, and pragmatic features such as application domain specification documents that help distributed users maintain consistent use of policy terms. In comparison with systems like SPKI/SDSI [10, 24] and KeyNote [4], the advantage of RT includes a declarative, logic-based semantic foundation also based on Datalog, support for vocabulary agreement, strongly-typed credential and policies, and more flexible delegation structures.

An entity in RT is a uniquely identified individual or process. They can issue credentials and make requests and we assume that one can determine which entity issued a particular credential or request. RT uses the notion of roles to represents attributes: an entity is a member of a role if and only if it has the attribute identified by the role.

In RT, a role is denoted by an entity followed by a role name, separated by a dot. Only the entity A has the authority to define the members of the role $A.R$, and A does so by issuing role-definition credentials. An entity A can define $A.R$ to contain $A.R_1$, another role defined by A . Such a credential reads $A.R \leftarrow A.R_1$; it means that A defines that R_1 dominates R . At the same time, a credential $A.R \leftarrow B.R$ is

a delegation from A to B of authority over R . This can be used to decentralize the user-role assignment. A credential of the form $A.R \leftarrow B.R_1$ can be used to define role-mapping across multiple organizations when they collaborate; it also represents a delegation from A to B . Using a linked role in a credential enables the issuer to delegate to each member of a role. The credential $A.R \leftarrow A.R_1.R_2$ states that: $A.R$ contains any $B.R_2$ if $A.R_1$ contains B .

To model permissions on objects and services one also uses roles. A permission typically consists of an access mode and an object. It is often useful to group logically related objects and access modes together and to give permission about them together. These groups are called o-set and are defined in ways similar to roles. The difference is that the members of o-set are objects that are not entities.

A o-set-definition credential is similar to a role-definition credential.

- $A.o(h_1, \dots, h_n) \leftarrow B.o_1(s_1, \dots, s_m)$
where $o(h_1, \dots, h_n)$ is an o-set name of base type τ , and $B.o_1(s_1, \dots, s_m)$ another o-set of base type τ .
- $A.o(h_1, \dots, h_n) \leftarrow A.r_1(t_1, \dots, t_l).o_1(s_1, \dots, s_m)$
where $o(h_1, \dots, h_n)$ is an o-set name of base type τ , and $A.r_1(t_1, \dots, t_l).o_1(s_1, \dots, s_m)$ is a linked o-set in which $r_1(t_1, \dots, t_l)$ is a role name and $o_1(s_1, \dots, s_m)$ is an o-set name of base type τ .

At present we do not have roles in our framework (through roles are present in the Tropos framework), and so, we do not translate role-definition credentials. The intuition is that $\text{permission}(ID, A, B, S1, S2)$ can be rewritten in RT framework as $A.S1 \leftarrow B.S2$, and $\text{delGrant}(ID, A, B, S1, S2, N)$ as $A.S1 \leftarrow B.r.S2$, where B allows to use the service $S1$ to actors in the role $B.r$.

The user of the system - patients, clinicians and administrative staff - are modeled as entity whose policies consists only of credentials they acquire over time.

Example 1. A patient allows his clinician to read his personal/medical data to provide accurate medical treatment. We express the trust relationship in our framework as $\text{permission}(id, Pat, Cli, Rec, MedTre) :- \text{isClinicianOf}(Pat, Cli) \wedge \text{owns}(Pat, Rec)$. The intuition is that $\text{isClinicianOf}(a, b)$ holds if the instance a is the clinician of the instance b . Now, we translate the relationship into the RT framework as

$\text{Pat.recordAc}(\text{read}, ?F : \text{Pat.record}) \leftarrow \text{Pat.clinician.provide}(?E : \text{medTre})$

Given “ $\text{Pat.record} \leftarrow \text{Rec}$ ” and “ $\text{Pat.clinician} \leftarrow \text{Cli}$ ”, one can conclude that “ $\text{Pat.recordAc}(\text{read}, \text{Rec}) \leftarrow \text{Cli.provide}(?E : \text{medTre})$ ”.

Example 2. The Medical Information System allows the clinician to write on his patient records to upgrade them. We express the trust relationship as $\text{permission}(id, MIS, Cli, Rec, \text{upgrade}(Rec)) :- \text{isClinicianOf}(Pat, Cli) \wedge \text{owns}(Pat, Rec)$

Now, we translate the relationship into the RT framework as

$\text{MIS.recordAc}(\text{write}, ?F : \text{Pat.record}) \leftarrow \text{Pat.clinician.upgrade}(?F : \text{Pat.record})$

Given “Pat.record \leftarrow Rec” and “Pat.clinician \leftarrow Cli”, one can conclude that “MIS.recordAc(write, Rec) \leftarrow Cli.upgrade(Rec)”.

8 Conclusion

The main contribution of this paper is the introduction of a framework that integrates security and requirements engineering. We have proposed a PKI/trust management requirements specification and analysis framework based on the clear separation of trust and delegation relationship. This distinction makes possible to capture the high-level security requirements without being immediately bogged down into considerations about cryptographic algorithms or security implementation. This should be similar to what happens when discussing functional system requirements: one doesn’t get immediately trapped into discussions about programming languages or Java patterns and coding techniques.

Further, the framework we proposed supports the automatic verification of security requirements specified in a formal modeling language. Particularly, we have used the DLV system to check system consistency. Finally, we have defined the trust management implementation of our framework into the RT framework.

The research developed here is still in progress. Much remains to be done to further refine the proposed framework and validate its usefulness with real case studies. We are currently working in the direction of incorporating explicitly roles adding time features and the integration with the Formal Tropos tool [11]. Also we are investigating the effects of supporting hierarchies of objects and hierarchies of actors.

References

- [1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [2] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Computer Publishing, 2001.
- [3] A. Bandara, E. Lupu, and A. Russo. Using Event Calculus to Formalise Policy Specification and Analysis. In *Proc. of the 4th Int. Workshop on Policies for Distributed Sys. and Networks (POLICY’03)*, 2003.
- [4] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The Role of Trust Management in Distributed Systems Security. *Secure Internet Programming*, 1603:185–210, 1999.
- [5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proc. of 1996 IEEE Symp. on Sec. and Privacy*, pages 164–173, 1996.
- [6] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An Agent-Oriented Software Development Methodology. *JAAMAS*, 8(3):203–236, 2004.
- [7] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss. REFEREE: Trust management for Web applications. *Computer Networks and ISDN Systems*, 29(8–13):953–964, 1997.
- [8] N. Damianou. *A Policy Framework for Management of Distributed Systems*. PhD thesis, University of London, 2002.
- [9] T. Dell’Armi, W. Faber, G. Ielpa, N. Leone, and G. Pfeifer. Aggregate Functions in Disjunctive Logic Programming: Semantics, Complexity, and Implementation in DLV. In *Proc. of IJCAI’03*. Morgan Kaufmann Publishers, 2003.

- [10] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. Simple Public Key Certificates. Internet Draft (work in progress), 1999.
- [11] A. Fuxman, L. Liu, M. Pistore, M. Roveri, and J. Mylopoulos. Specifying and analyzing early requirements: Some experimental results. In *Proc. of RE'03*, page 105. IEEE Press, 2003.
- [12] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning. In *Proc. of iTrust 2004*, volume 2995 of *LNCS*, pages 176–190. Springer-Verlag Heidelberg, 2004.
- [13] S. Guttorm. Eliciting Security Requirements by Misuse Cases. In *Proc. of TOOLS Pacific 2000*, 2000.
- [14] T. Jim. SD3: a trust management system with certified evaluation. In *Proc. of 2001 IEEE Symp. on Sec. and Privacy*, pages 106 – 115. IEEE Press, 2001.
- [15] J. Jürjens. Towards Secure Systems Development with UMLsec. In *Proc. of FASE'01*, 2001.
- [16] N. Li, B. N. Grosz, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *TISSEC*, 6(1):128–171, 2003.
- [17] N. Li and J. C. Mitchell. Datalog with Constraints: A Foundation for Trust-management Languages. In *Proc. of PADL'03*, 2003.
- [18] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of A Role-based Trust-management Framework. In *Proc. of 2002 IEEE Symp. on Sec. and Privacy*, 2002.
- [19] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed Credential Chain Discovery in Trust Management. *J. of Comp. Sec.*, 11(1):35–86, 2003.
- [20] L. Liu, E. S. K. Yu, and J. Mylopoulos. Security and Privacy Requirements Analysis within a Social Setting. In *Proc. of RE'03*, pages 151–161, 2003.
- [21] T. Lodderstedt, D. Basin, and J. Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In J.-M. Jezequel, H. Hussmann, and S. Cook, editors, *Proc. of UML'02*, volume 2460, pages 426–441. Springer-Verlag Heidelberg, 2002.
- [22] J. Lopez, A. Mana, J. A. Montenegro, and J. J. Ortega. PKI design based on the use of on-line certification authorities. *Int. J. of Information Sec.*, 2(2):91 – 102, 2004.
- [23] J. McDermott and C. Fox. Using Abuse Case Models for Security Requirements Analysis. In *Proc. of ACSAC'99*, 1999.
- [24] R. L. Rivest and B. Lampson. SDSI – A Simple Distributed Security Infrastructure, 1996.
- [25] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice-Hall, Englewood Cliffs, New Jersey, 1999.
- [26] T. Tryfonas, E. Kiountouzis, and A. Poulymenakou. Embedding security practices in contemporary information systems development approaches. *Inform. Management and Comp. Sec.*, 9:183–197, 2001.

A Framework for Evaluating the Usability and the Utility of PKI-enabled Applications

Tobias Straub¹ and Harald Baier²

¹ Darmstadt University of Technology, Computer Science Departement
D-64283 Darmstadt, Germany
`tstraub@gkec.tu-darmstadt.de`

² Darmstadt University of Technology, Darmstadt Centre of IT Security (DZI)
D-64283 Darmstadt, Germany

Abstract. Besides the pure technical features, the usability of a PKI-enabled application plays a crucial role since the best security application will fail in practice if its usability is insufficient.

We present a generic framework to evaluate the usability and utility of PKI-enabled applications with respect to their security features. Our approach is modeled on the Common Criteria methodology and consists of 15 evaluation categories that cover all the relevant topics, namely deployment, ergonomics, and technical features.

Keywords: Common Criteria, cryptography, ergonomics, evaluation, PKI, usability, usefulness, utility

1 Introduction

A public key infrastructure (PKI) based on key pairs and digital certificates is the fundamental architecture to enable the use of public key cryptography. A lot of today's commercial off-the-shelf (COTS) software has already implemented some form of PKI support, consider e.g. email clients, web browsers, file encryption software, groupware, VPN clients to name only a few. These so-called *PKI-enabled* applications are the focus of attention in this paper.

In order to achieve the security goals authenticity, integrity, and confidentiality, a PKI-enabled application has not only to provide an appropriate implementation, but also a sufficient level of usability. The user *must* be able to make use of the security features in practice, otherwise the best security application will fail. The last statement is due to the fact that users' behaviour is often the biggest risk in security software. This is confirmed by a survey prepared by the KPMG consulting group in cooperation with the security magazine *kes* [12]. Incidents caused by users are at the top of the list (52%) of security threats. To

¹ The author's work was supported by the German National Science Foundation (DFG) as part of the PhD program "Enabling Technologies for Electronic Commerce" at the Darmstadt University of Technology.

draw a comparison, malware like viruses, worms, or Trojan horses is responsible for only 25% of the security incidents. However, both areas are often linked together.

The present paper grew out of a usability study to evaluate how applications can be secured using PKI [2]. We carried out this project in 2003 by the order of Microsoft Deutschland GmbH, examining a total of 38 different products ranging from standard applications (email client/server, web browser/server, groupware), access control tools for local and remote login (WLAN, VPN) to CA software and cryptographic APIs for application developers.

The main contribution of the paper at hand is the development of a generic framework to evaluate the usability and utility of PKI-enabled applications. An evaluation yields a numeric score which can be weighted to fit the actual priorities and external factors like the intended use case or budget constraints. In our opinion, such a score offers a good basis for making a decision in favour of a particular PKI-enabled application. The framework may as well be useful as a requirements specification for application designers. We are not aware of any comparable framework for evaluating PKI-enabled applications.

2 Related Work

In this section, we will briefly familiarize the reader with the subject of usability in the context of security. We use a taxonomy which is due to Nielsen [9]: *Usability* and *utility* are considered common sub-categories of the category *usefulness*. Usefulness and other characteristics like costs or reliability are aspects of a system's *practical acceptance*. The question whether a system, in principle, has the functionality necessary for a certain task is a matter of utility, whereas usability refers to how this functionality can be used in practice.

The design of human-computer interfaces in general has been studied extensively (see e.g. [9] for an introduction). However, this does not hold for the special case of security applications, not to mention PKI-enabled software. Research in this field has been somewhat focused on the usage of password-based security mechanisms since they are a wide-spread authentication method (see e.g. [10, 1] for a survey).

It has become a common belief that users' behaviour is often the biggest security risk [12, 11]. Nevertheless, a paradigm shift towards the *user-centered* design approaches, which is urgently needed, has failed to appear [5, 1]. As Davis [3] pointed out, users have to pay a certain price for the technological benefits of public-key compared to symmetric-key cryptography. Systems using public-key cryptography transfer responsibilities to the users that are otherwise being centrally handled by a server or administrator. Among these burdens are the management of keys, certificates, and status information.

An empirical evaluation of a PGP-capable email client confirmed the assumption that this is in fact too hard to handle for average users. Whitten and Tygar [13] showed that in practice this software does not achieve a sufficient level of security due to its usability deficiencies. Their paper also identifies a number

of security-inherent properties which are listed below. These properties need to be addressed by an implementation; we will often refer to them in Section 3.

- (P1) Unmotivated user property** Since security is hardly ever a user's primary goal, he cannot be expected to pay too much attention to it or, for instance, go through voluminous manuals before using the software.
- (P2) Abstraction property** Especially in the field of public-key cryptography and PKI, it is a difficult task to find intelligible yet precise (real world) metaphors for abstract mathematical objects like key pairs or certificates.
- (P3) Lack of feedback property** Applications should above all enforce security in a seamless way. In cases where interaction is required, the user should be provided with the necessary feedback to allow for an appropriate decision. Presenting a complicated matter in a short dialogue is challenging.
- (P4) Barn door property** Dangerous and irrevocable mistakes must be ruled out from the start. If a secret has been revealed at a certain point of time, say during an insecure network connection, one cannot say whether an attacker might already know it.
- (P5) Weakest link property** Security is considered to be a chain the weakest link of which determines the strength of the whole system. As a consequence, users need to comprehensively take care of an application's security settings.

Several standards define guidelines for ergonomic user interfaces (UI) in general, for instance ISO 9241 [7]. Taking the before-mentioned properties into account, an adjustment of the design principles to the context of security software is necessary [4]. The relevance of the criteria of ISO 9241 Part 10 ("Dialogue principles") for security software is outlined in Figure 1. The modified principles can serve both as a practical guideline for UI designers and for evaluation purposes. Our framework presented in Section 3 will also refer to these principles sometimes.

Another usability evaluation framework is due to Kaiser and Reichenbach [8]. It classifies different types of usability and security problems taking user errors and their security relevance as an indicator. This framework distinguishes between problems that can be traced back to a user's lack of security competence and problems which arise despite a user's skills. The motivation to do so is the observation that these two problem classes need different treatments during a redesign process of the software.

A very important framework for the evaluation of IT products and systems with respect to their security mechanisms are the Common Criteria [6] which are a result of merging standards from several European countries (ITSEC), the U.S. ("Orange Book"), and Canada. Despite their complexity, usability has only marginal relevance in the Common Criteria.

3 The Framework

In this section, we present our framework to measure the usability and utility of PKI-enabled applications with respect to their security features. The rating process resembles that of evaluating security products as proposed in the Common

general principle	relevance for security applications
error tolerance	Error prevention is considered the most important principle because of P4 and P5.
suitability for individualisation, controllability	Users should be guided more rigorously where needed to avoid weaknesses due to misconfiguration or mistakes (P4).
suitability for learning	Humans tend towards the "trial and error" method when using an application for the first time (P1). This must not lead to severe errors.
self-descriptiveness	Presenting security mechanisms and error messages clearly and precisely is crucial (P2, P3).
conformity to user expectations, consistency	The application should use a correct, homogeneous and usual terminology to describe things (P3).
suitability for the task	Users want to accomplish security-related tasks easily and efficiently. The design should take into account that users have varying abilities (P1).

Fig. 1. UI design principles for security applications (partially based on [4]).

Criteria [6]. Since the Common Criteria methodology is an ISO standard and receives worldwide acceptance, we consider it a reasonable basis for our approach, too. Throughout our framework we will often use the term *user*. Depending on the concrete scenario, this may either denote an end-user or an administrator.

3.1 Evaluation Categories

In all, our framework consists of 15 evaluation categories which are evenly spread in three groups named *deployment issues*, *ergonomics*, and *security features*. The motivation behind this choice is Schneier's famous saying "security is a process, not a product" [11]. We thus have to follow an integral approach and must not restrict ourselves solely to ergonomics or technical features. To illustrate the ideas behind each category and to simplify the evaluator's task, we propose a number of exemplary questions. These *central questions* have to be investigated throughout the evaluation to obtain a category score.

In order to make use of a PKI-enabled product, it has to be deployed. For this reason, the first category group is named *deployment issues*. For instance, the user has to obtain the product and install its components which often means that he has to be familiarized with the PKI features before being able to use them. We turn in detail to the deployment issues in Section 3.3. With property P1 in mind, this category should not be underestimated.

Second, the handling of a PKI-enabled application in service is discussed in the category group *ergonomics*. In a classical sense this category group is the core of any usability. Typical categories of ergonomics are the composition of the menus and dialogues, respectively, and the help system available. We present the central questions of ergonomics in Section 3.4.

The categories defined so far may easily be adapted to other use cases than PKI. However, the best product with respect to ergonomics is pointless, if the use case specific security features are substandard. Our third category group *security features* takes the properties P4 and P5 into account. It deals with the algorithms and key lengths in use, the certificate management, and how the validation of a certificate may be done. We turn to the security features in Section 3.5.

3.2 Usability Profiles

In order to obtain a quantitative measure, an evaluator will award points for each category. The result of each category is an integer score in the interval $[-2, 2]$ where 0 denotes an average result, ± 1 a little above/below average result and ± 2 an outstanding/substandard result. The guidelines on how to assign points are part of a so-called *usability profile*. A usability profile has to be defined before the evaluation actually takes place. This can be done based on the central questions of our framework. Each usability profile has to list sample answers and a guideline on how to award points. We provide appropriate clues in the following sections.

The definition of the usability profile may for instance be done by the evaluator himself or by a usability specialist involved in the process. An evaluation result has always to be interpreted with respect to the underlying usability profile. We point out that this procedure is analogous to the definition of so-called protection profiles as described in the Common Criteria.

Our framework allows for an adaptation to different environments since the partial results of a category group are weighted to get an overall score. The weight definition depends on the concrete usage scenario and is part of the usability profile. Figure 2 shows three coarse examples for scenarios with *high security* requirements and skilled personnel (e.g. in the military or a research department), *enterprise* usage with low deployment and helpdesk budgets (with average security demands), and a *private* usage scenario. Let us quickly explain the proposed weighting in Figure 2 in case of private usage: A private user is in general not supported by an administrator, that is he has to deploy the PKI-enabled product himself. In addition, the unmotivated user property P1 requires an easy to deploy and easy to install application. Therefore, deployment is rather important which is why we assign a weighting of 40% to this category group. The

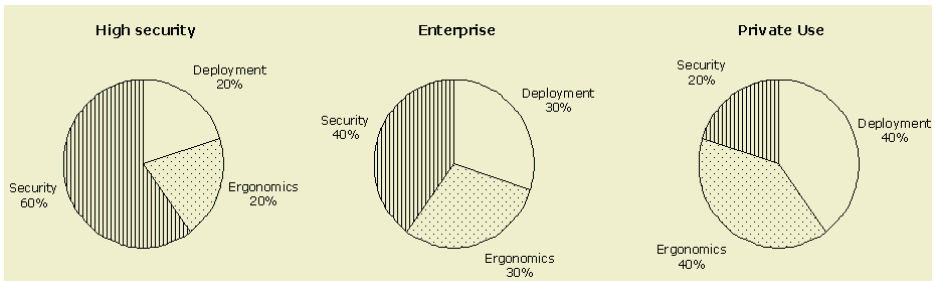


Fig. 2. Sample weightings of the category groups.

properties P2, P3, and P4 motivate a weighting of 40% for the group ergonomics, too. Due to the small assets and threats in this scenario, the security features are of secondary importance and thus weighted by 20%.

3.3 Deployment Issues

We first turn to the category group *deployment issues* which describes the pre-arrangements necessary to set up the system. The central questions listed in the following mainly address property P1, i.e. the user's little motivation to really use built-in security features of existing software or to change to another application of the same kind for the sake of better support of PKI features. Suppose that a computer administrator or an end-user chooses to increase the level of enterprise-wide or personal security, respectively. Then it should be made as easy as possible to realize this plan. People who once tried to improve security but did not succeed are likely to get frustrated and to give up.

In all, the deployment issues comprise the following categories: The first step is *gathering information* about candidate products which is closely related to the question how the software itself can be *obtained*. Since sophisticated applications tend to have special infrastructure demands, it is necessary to consider *technical requirements* before *installing and configuring* the application. During the latter two steps, there may arise the need for technical *support*. Finally, it should be assessed how much *training* is required for the product to be used securely. We present in detail the central questions for each category in what follows.

Gathering Information and Obtaining the Software If an evaluator has no a priori knowledge of appropriate products, it is important that comprehensive information, maybe even an evaluation version, of the software is easily available. In addition, analysis from professional journals or experience reports found in news groups provide more neutral and objective information. It goes without saying that the easier information and the product itself are available, the better the rating is in this category. Central questions are:

- How much meta information (surveys, tests etc.) comparing different products of the same kind is available?
- Is there already enough experience about a product of recent date?
- Does the vendor offer precise product information on his web pages?
- How complicated is it to obtain the product, can it be purchased online?
- Is it possible to upgrade an evaluation copy to a full version simply by an activation code, i.e. without re-installing and re-configuring the software?

Technical Requirements A successful installation often depends on the technical environment in use. For instance, the hardware of a five-year old PC is not sufficient to run current server applications due to performance and memory restrictions. Again, we refer to the unmotivated user property and stress that a good PKI-enabled application should make it easy for users getting started with

it. It should ideally not require high-end hardware nor additional software (e.g. special plugins) to do its task properly. These requirements can only be rated in the light of the actual prerequisites, since some software, e.g. for a certification authority, may in fact have very particular needs. We thus propose that the easier it is for the user to fulfill the requirements of the software, the better the application is rated. In this context, "easier" has to be interpreted in terms of time or money.

- Is it possible to install and operate the application with existing hard- and software?
- Are multiple operating systems supported? Which ones?
- Does the installation require additional programs?
- If the application needs additional hard- or software: Which/how many third-party products are supported?

Installation and Configuration The background of this category is that proper installation and initial configuration are crucial to security with respect to the barn door property P4. On the one hand, there should be enough support, e.g. an installation wizard or a list of commonly used configurations, to guide the user. On the other hand, the default configuration must provide an appropriate security level. This is due to the consideration that it may be a security novice who is installing the software and who is above all not interested in the security features themselves but in getting the software running. An example is the selection of cryptographic parameters like key sizes. While this does not affect the functionality of the software, it is indeed security-relevant.

- Is there an installation wizard guiding an inexperienced user?
- Does the product offer to choose from predefined scenarios?
- How difficult is it to adapt the configuration to individual needs?
- How does the software check user settings for consistency and security?
- Which level of security does the default configuration provide?
- How much expertise is necessary to successfully install and configure the application for the first time?

Technical Support Even if the PKI-enabled software may have advanced installation and configuration wizards, a user may sooner or later need additional help. Therefore, we direct our attention to the quality of support, too. Once again, the criteria in this category strongly depend on the actual application scenario. The rating scheme is analogous to that of the category *technical requirements*.

- What kind of support does the vendor (the supplier) offer? How expensive is that support? Is it charged per request or is there an annual fee?
- How can the user contact the support team? (e.g. phone, email) At what times? (7x24 or office hours only)
- Are there FAQ or manual pages on the Internet?
- Is the information provided in the user's native language?

Training The deployment of every new application should ideally be accompanied by training lessons for users. As we all know, this is more wishful thinking than reality. Due to the abstraction property P2, teaching users to deal with security-related tasks may in particular gain importance where ergonomic deficiencies have to be compensated.

- Which amount of training for the intended users is absolutely necessary to provide an appropriate usage?
- Does the software provide an interactive guided tour explaining its basic functionality?
- How many companies do offer training lessons? At what price?
- Which books do provide course materials?

3.4 Ergonomics

One may argue that *deployment issues*, which we have discussed in the previous section, also fall into the range of usability. To make clear the following difference, we decided to name the current section "ergonomics": Contrary to *deployment issues* which cover non-recurring actions during the setup process, *software ergonomics* is an everyday concern which in general affects much more tasks and people. It may be tolerable that the installation phase requires expert knowledge (which means that the category group *deployment issues* is assigned a smaller weight in the usability profile). But this does not hold for the "operational" phase where people are expected to actively use the PKI-enabled application to get their work securely done.

A natural category to start with is the UI design concerning *menus and dialogues*. The information given there should be self-descriptive using a consistent terminology, whereas its organization should try to anticipate how users will operate the application (cf. Figure 1). Because of property P1, security software should follow the maxim to hide complexity from the average user, acting in a seamless way. Hence, the second category is the *transparency of security features*. Since it cannot be guaranteed a hundred percent that the respective PKI-enabled application will always work without interaction, *warning and error messages* are treated in the corresponding section. It is exactly in this situation when a user may question the *help system*, probably for the first time (à propos "trial and error", cf. Figure 1). From the viewpoint of security, it all depends on how the application *reacts in potentially threatening situations* since errors should be prevented at all costs.

Menus and Dialogues This category groups two types of UI components, the menus being the more static and dialogues being the more dynamic elements. Once again, we think of the user as a security novice and unmotivated subject (P1). Therefore, on the one hand it should be possible to intuitively find *all* the security-relevant menu entries because of P5. On the other hand, dialogues should show a careful design due to the lack of feedback property. Where users

are supposed to change security parameters, there should be clues to underline sensitive settings and a possibility to restore a previous or even a default configuration. A general requirement is that technical terms stemming from PKI and cryptography should be used in a precise and consistent manner.

- Does the organization and structure of PKI-relevant menus follow the general principles used throughout the application?
- How are the security-relevant menu items organized? Are they easy to find and grouped reasonably?
- Does the visual design of the dialogue reflect its level of security relevance? (e.g. special symbols for low, medium, high security relevance)
- Are the technical terms chosen well and intelligible? (even more important if the application does not come in its native language)

Transparency of the Security Features This category refers to a common paradigm to encounter the unmotivated user property. UI experts agree that an ordinary user should at first get into contact with as few security features and settings as possible. However, the more he is interested in the details, the more information the application should provide. This serves a twofold purpose: First, to increase a user's trust in the application by giving further insights. A second aspect is the suitability for learning (cf. Figure 1). A standard technique, for instance, to both hide complexity and allow users to get deeper into it, is an "Advanced Settings" button.

- To what extent does the user have to get involved in security settings to be able to use the features?
- Do the default settings meet the requirements of an ordinary user?
- How difficult is it to change the transparency settings?
- Can a user adjust the transparency level according to his individual skills? (e.g. ordinary user, interested layperson, expert)
- Is every necessary piece of information accessible at all? (e.g. details about cryptographic algorithms and protocols)

Warning and Error Messages In situations where the user or his communication partner behaves wrong or in a non-standard way, this category plays an important role. A user's decision when confronted with a warning or error message is often crucial for security. The user must understand the warnings and behave in an appropriate way. Otherwise he will just click the "close" or the "continue" button. It is therefore indispensable that an explanation of alternative actions and their consequences is given. A good practice is not to force the user to make an immediate decision, which is the case when using modal dialogue windows, but to give him the possibility of proceeding in another way, e.g. browsing the help pages or postponing the task. Consequently, the abstraction property, the barn door property, and the weakest link property are the basis for the following central questions.

- How many details are given in message dialogues?
- Does the text explain what to improve if a single step of a complex PKI operation, e.g. signature verification, fails?
- Are there warnings in case of a security-sensitive operation? What advice is given in such a situation? (e.g. if the chosen key lengths are too small)
- How precise, understandable, and insistent is the information given? Does the message encourage to ignore it?
- Are there useful links to matching topics in the help system?

Help System The built-in help system of an application is the primary source to get information during usage. The help system should be designed to suffice an average user in most of the cases. We think it advisable to have a short glossary of commonly used technical terms of cryptography and PKI, as well as a short introduction to the topic on a rather easily intelligible level, but with links to more detailed information. A nice feature would for example be an integrated FAQ section. While it is common practice to refer the user to the vendor's web site for further help using hyperlinks in dialogues, we vote against it. For the sake of convenience this mechanism should only be used for non-critical, additional, and up-to-date information. But not for answering standard requests since a user may not be connected to the Internet at the moment a question arises. We consider it a must to have context-sensitive help in all security-related menus and dialogues, at least in the most critical ones. The central questions touch the same areas (P2, P4, P5) in the current category than in the previous one.

- Which of the main technical terms of PKI and cryptography are explained? (e.g. certificate, key pair, certification authority) Are the explanations correct and intelligible?
- How detailed is the introduction to PKI and its concepts? Is the text written in an abstract fashion or are there useful examples?
- Is a context-sensitive direct help implemented? If yes, in which way does it support the user? Are there further links to the help system?
- In which way is the user assisted when accomplishing a complex task? (e.g. when installing his own private key)
- Are there links to external information sources about PKI and current developments?

Reaction in Potentially Threatening Situations In order to get a reasonable measure on the utility of a PKI-enabled application, we have to test its reaction in critical situations. We consider this worth a separate category due to the potentially high damage that may be caused in case of failure. A typical scenario is a digitally signed message which has been tampered with. Then the application has to reject the document and explain in detail not only the reason of failure, but give a concrete guidance which measures the user has to take next. In case the local key or certificate store is manipulated, the application must react accordingly and trigger an alert. It is also important how status information is handled. Suppose, the application wants to use an X.509 certificate

which specifies the URL of a CRL distribution point or an OCSP responder, but the corresponding server is unreachable.

- What is the behaviour of the application in case of a certificate that cannot be validated successfully due to an unknown issuer?
- Does the application tell the user how to check the fingerprint of a certificate when introducing a new CA to the system?
- How does the program react in case when a source of status information specified in the certificate is unreachable?
- Does the application has the ability to automatically check for and download security updates and patches?

3.5 Security Features

The subject of the third and last category group are the security features of the PKI-enabled application under evaluation. These features relate indirectly to the issue of usability as decisions on the technical layer often have an influence on UI design. If, for instance, the application does not implement functionality which is necessary for using PKI in a certain environment or does not support a common standard, users are urged to find a way to work around this problem (probably weakening the system). Security on the cryptographic layer and utility of a PKI-enabled product are thus crucial factors for user acceptance, too. With regard to properties P4 and P5, we emphasize that the application should provide a proper implementation of the security mechanisms since it is virtually impossible for the user to detect flaws on this layer.

The arrangement of the following categories is relatively straight-forward: First of all, we review *algorithms and parameters* which belong to the cryptographic layer. The next category revolves around the *handling of secret keys*. This will lead us to the *certificate management* and how the application deals with *status information*. A last category is named *advanced functionality*.

Algorithms and Parameters In order to achieve compatibility towards different communication partners, an appropriate set of cryptographic algorithms should be available. We emphasize that a reasonable selection of methods should include standard and widely deployed algorithms. This does not call for as many as possible and moreover exotic algorithms as this may bear additional risks (at least to confuse the user). However, it is important that the cryptographic building blocks rely on independent mathematical problems, e.g. integer factorisation and discrete logarithm in an elliptic curve (EC) group. This is because of the observation that mathematical or technical progress may render a single algorithm insecure. The cryptography inside the application cannot be treated separately since its effectiveness also depends on the protocols built on top of it. Thus, the word algorithms extends to protocols in this section. The second aspect in this context are the cryptographic parameters, typically including key sizes and lengths of message digests.

- How many different combinations of cryptographic algorithms and parameters are available?
- Does the selection comprise state-of-the-art algorithms? (e.g. AES, RIPEMD, EC cryptography)
- Is it possible to choose individually from the set of combinations of algorithms and parameters? Does the system warn against weak combinations?
- Are all weak combinations disabled in the default settings?
- Are cryptographic primitives based on independent mathematical problems?
- Does the application implement a strong pseudo-random number generator? What source of random is used as seed?

Secret Key Handling This category is about how secret keys are generated, protected, applied, transferred, and destroyed by the application. In this paragraph, we use the term secret key in a generic way. While it is often replaced by the term private key in the context of a public key scheme, a secret key in a narrower sense is used for a symmetric algorithm, e.g. as an ephemeral session key. It goes without saying, that secret keys have to be protected carefully by versatile technical precautions.

- Is the application able to generate keys and key pairs on behalf of the user?
- What kind of storage devices for secret keys are supported? (e.g. smart card with corresponding reader, USB token, HSM, or soft token)
- Which cryptographic and other technical precautions are provided to secure a secret key when stored on the hard disc or in memory?
- How is a secret key enabled for usage? Is it possible to enable a batch processing? (e.g. for bulk signature generation)
- In case a secret key is exportable: Are there efforts to enforce a certain password complexity when using password-based encryption?
- Can key data be securely destroyed, e.g. by multiply over-writing the corresponding area on the hard disc?

Certificate Management We now turn to the management of third party certificates. Two things are important in this context: In our opinion, the application should first support the import of certificates via different channels to facilitate this task for the user. This means that a lookup on a certificate server or – in the PGP terminology – *key server* should be supported, using different protocols like LDAP, HTTP, FTP, or simply retrieving the certificate from the file system. The second question is how the integrity of the certificate database is guaranteed. If the certificates are stored on the hard disc, a typical method is to use a password-based MAC or a central storage location to ensure that the database has not been tampered with.

- How does the import and retrieval of certificates take place?
- Is it possible to read/write a certificate from/to a hard token?
- Does the application cope with certificate trust lists (CTL) for import or export?

- Which technical measures are taken to protect the certificate store?
- How can the user assign and withdraw trust to third party certificates? Is this possible on a fine-grained basis?
- Can the certificate handling be delegated to a central point or authority? (e.g. the OS or an administrator)

Status Information Especially with regard to signature verification in the context of non-repudiation, the ability to gather information about the validity of a certificate at a certain point of time is crucial. It is also important to guarantee a timely revocation of public keys used for encryption. However, status information often is not retrieved automatically or processed properly by PKI-enabled applications. As of today, different methods are available to answer a validation request. We point out that the answer to such a request is nontrivial, as there are different validity models.

- Does the PKI-enabled application automatically gather status information?
- Which mechanisms for status information are supported? (e.g. CRL, delta CRL, indirect CRL, online requests via OCSP)
- Does the application make use of a CRL distribution point entry in an X.509 certificate?
- Does the application allow to manually import status information obtained from another source?
- Is there a mechanism to archive validation material?
- Does the application ensure that the machine's local time is correct?

Advanced Functionality Finally, we come up with some additional features. While some of these features are well known in the community, they are often not implemented by current PKI-enabled applications. Particular use cases, like for instance the processing of signed documents which is subject to digital signature legislation, may have very special requirements. In environments with high security demands, e.g. for governmental use, a certain level of security evaluation may be an absolute must.

- May an administrator enforce a group policy concerning e.g. key export or maximal usage periods for cryptographic keys?
- Is the application ready for enhanced concepts like cross-certification or bridge CAs? Are time stamping services supported?
- In case the application processes X.509 certificates: Which extensions are implemented, are critical extensions always handled correctly? Does the application require proprietary extensions? Is the application compliant to major standards? (e.g. the PKIX or ISIS-MTT profile)
- Does the application support key sharing/backup? (e.g. à la Shamir)
- Has the application undergone an evaluation of its security features? (e.g. according to the Common Criteria)

4 Conclusions

We have presented a new and generic framework to evaluate the usefulness of PKI-enabled applications. It is not restricted solely to usability, but also treats deployment issues and security features to get utility ratings, too. Our framework can be adapted to a multitude of PKI use cases by means of a usability profile. Due to space restrictions, we could not give a sample profile here.

We designed this framework having the high PKI-inherent complexity in mind which requires special attention. However, the framework could be extended to cover general security applications as well.

Acknowledgements

We received helpful critique from the anonymous reviewers. We also thank Gerold Hübner (Microsoft Deutschland GmbH) for fruitful discussions on the topic.

References

- [1] A. Adams and M. A. Sasse. Users are Not the Enemy: Why Users Compromise Security Mechanisms and How to Take Remedial Measures. *Communications of the ACM* 42 (12), 1999, 41–46.
- [2] J. Buchmann, H. Baier, and T. Straub. Absicherung von Anwendungen mit der Unterstützung von Public-Key-Infrastrukturen – Benutzbarkeitsstudie im Auftrag der Microsoft Deutschland GmbH, 2003. (in German)
- [3] D. Davis. Compliance Defects in Public-Key Cryptography. *6th USENIX Security Symposium*, San Jose, USA, 1996.
- [4] D. Gerd tom Markotten and J. Kaiser. Usable Security – challenges and model for e-commerce systems, *Wirtschaftsinformatik* (6), 2000, 531–538.
- [5] U. Holmström. User-centered design of security software. *17th International Symposium on Human Factors in Telecommunications*, Copenhagen, Denmark, 1999.
- [6] ISO 15408: Common Criteria for Information Technology Security Evaluation (CC) Version 2.0, 1998.
- [7] ISO 9241: Ergonomic requirements for office work with visual display terminals.
- [8] J. Kaiser and M. Reichenbach. Evaluating security tools towards usable security. *IFIP 17th World Computer Congress*, Montreal, Canada, 2002.
- [9] J. Nielsen. Usability Engineering. AP Professional, Cambridge, 1993.
- [10] M. A. Sasse. Computer Security: Anatomy of a Usability Disaster, and a Plan for Recovery. *CHI 2003, Workshop on Human-Computer Interaction and Security Systems*, Fort Lauderdale, USA, 2003.
- [11] B. Schneier. Secrets and Lies, *Wiley*, 2000.
- [12] J. Voßbein and R. Voßbein. KES/KPMG-Sicherheitsstudie: Lagebericht zur IT-Sicherheit. *kes* 3 and 4, 2002, available online <http://www.kes.info>. (in German)
- [13] A. Whitten and J. D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. *8th USENIX Security Symposium*, Washington DC, USA, 1999.

Using LDAP Directories for Management of PKI Processes

Vangelis Karatsiolis^{1,2}, Marcus Lippert¹, and Alexander Wiesmaier¹

¹ Technische Universität Darmstadt, Department of Computer Science,
Hochschulstraße 10, D-64289 Darmstadt, Germany
{karatsio,mal,wiesmaie}@cdc.informatik.tu-darmstadt.de

² Fraunhofer Institute for Secure Telecooperation,
Dolivostr. 15, D-64293 Darmstadt, Germany
karatsio@sit.fhg.de

Abstract. We present a framework for extending the functionality of LDAP servers from their typical use as a public directory in public key infrastructures. In this framework the LDAP servers are used for administering infrastructure processes. One application of this framework is a method for providing proof-of-possession, especially in the case of encryption keys. Another one is the secure delivery of software personal security environments.

Keywords: Public Key Infrastructure, PKI, Proof-of-Possession, PoP, Directory Services, LDAP, Personal Security Environment, PSE.

1 Introduction

A Public Key Infrastructure (PKI) provides practical use of public key cryptography proposed by Diffie and Hellmann [DH76]. Entities register for the services of the PKI in order for them to achieve security goals like authenticity and confidentiality using digital signatures and encryption. One of the most typical products of a PKI are digital certificates. They bind a public key to an entity which owns³ the corresponding private key. Commonly met certificates follow the X.509 [X.509] and the RFC 3280 [HPFS02] standards. A PKI is usually composed of two components namely the Registration Authority (RA) and the Certification Authority (CA).

The task of the RA is to register an end-entity in the PKI by examining its credentials. This entity may either already possess a key pair or not. In the first case, the public key is provided by the entity to the RA during registration and forwarded to the CA that issues the certificate. In the second case, the key pair is produced by the CA and, together with the issued certificate, delivered to the entity. In either case, the CA has to guarantee that finally the certificate is issued for the entity that owns the corresponding private key.

³ An entity is considered to be the owner of a private key if and only if it can legally use it.

The problem that arises in the first case is for the CA to determine whether this entity does really own the corresponding private key or not. Therefore, the entity must provide a Proof-of-Possession (PoP) for this private key. In several certification request syntaxes special fields are reserved for this purpose. An important issue with PoP is the number of messages needed for accomplishing it. While for signature keys the straightforward solution is to provide a signature during registration, the situation is not so easy for encryption keys. One good solution in terms of number of messages is the indirect scheme where the issued certificate is encrypted with the included public key and thus can only be read by the intended recipient. We provide an easy and efficient variant of this scheme using a directory based on the lightweight directory access protocol (LDAP) [HM02]. Furthermore, it can be extended to tightly link the PoP to the registration procedure. Moreover, a successful PoP is necessary for activation of the certificate. Whether or when this activation will take place is under full control of the entity.

In the second case, the problem is to securely deliver the private key to the registered and identified entity. A commonly used standard for software personal security environments (PSE) is PKCS#12 [PKCS12]. Usually, such PSEs are either handed out face to face or sent by e-mail. For automated certification processes in which management should be simple, the solution of physical presence can not be used since it hampers the automated process and requires human administration. Sending the PKCS#12 file by e-mail is secure since PKCS#12 has mechanisms to ensure privacy and integrity. Nevertheless, the e-mail may be intercepted by a third party which can try to extract the keys (in a password protected PKCS#12 file this may be easy for a weak password). Moreover, the e-mail may never reach his recipient due to an incorrectly configured spam filter or even a mailbox which has reach its capacity limit. We suggest a scheme based on an LDAP directory to ensure that only the legitimate recipient will receive his PSE without any eavesdropping from a third party.

This paper is organized as follows: In Section 2, we discuss specifications on certification request messages and examine their proposals for achieving PoP. In Section 3, we describe our proposed schemes and argue on their usability. Lastly, in Section 4 we conclude the paper and discuss future work in this direction.

2 Certification Request Messages and PoP

Common practice for the registration of end-entities in a PKI environment are special certification request message syntaxes. Several specifications address this problem.

The most commonly used one is the PKCS#10 [PKCS10]. It defines a syntax in which entities can provide information needed for creating a certificate from the CA. Furthermore, it enables them to include other data that can be used during and after the certification process. This syntax requires the message to be secured with a digital signature. Among other purposes, this is done to provide

a PoP for the corresponding private key, since the signature can be verified from the receiving party (RA or CA). However, this syntax does not consider keys used for encryption only. Such keys might have to be treated differently due to limitations of the cryptosystem or different security requirements.

Another specification is the Certificate Request Message Format (CRMF) [MASK99]. While handling signature keys identically to PKCS#10, it provides three methods for PoP of encryption keys. The first is to reveal the private key to the CA. The second one, called direct method, requires exchange of challenge-response messages and thus needs extra messages. The third one, the indirect method, is to encrypt the issued certificate with the contained public key and have the end-entity to send the decrypted certificate back in a confirmation message and by this demonstrate ownership of the corresponding private key.

Similar proposals can be found in Certificate Management Protocol (CMP) [AF99]. Special care about encryption keys is taken also in Certificate Management Messages over CMS (CMC) [MLSW00]. CMC specifies a mechanism for PoP which requires more than a single-round trip⁴ and therefore complicates the process. Lastly, XKMS [XKMS] considers the PoP problem exclusively for signature keys. For a discussion on PoP see also [ANL03].

The above proposals do not provide satisfactory solutions in cases where the end-entity does not want to reveal its key or the certification process management should be kept minimal and simple.

We believe that PoP is of great importance in the case of encryption keys. A certificate containing a public key for which the corresponding private key is not owned by the intended end-entity or does not exist at all is actually unusable. Any message encrypted with this public key can not be decrypted by the end-entity. This scenario can become extremely dangerous in cases where the original message is destroyed⁵ and only the encrypted message exists. Therefore, no one should be able to use a certificate containing encryption keys without prior acknowledgement of its owner.

To solve the PoP problem we propose a scheme similar to the indirect method described above, in which an end-entity is an authenticated user of an LDAP directory, which he can download and activate his certificate from. But in order to do so, he must decrypt a secret (implicit use of his private key), decrypt the certificate and then put it back on the LDAP server. Thus, we can avoid extra confirmation messages to the CA.

3 The New Schemes

3.1 LDAP Basics

LDAP is a protocol for providing access to directories based on the X.500 specification [X.500]. The current version is LDAPv3 specified in [HM02]. Internet

⁴ A request which is done from the end-entity, processed from the CA and returned to the end-entity.

⁵ This is a typical function of many encryption clients.

directories speaking LDAP are most commonly used in PKIs for dissemination of public information. It is exactly this “public directory” mentioned in [DH76] when public key cryptography was discovered. LDAP directories serve as the place where clients can download certificates of other users in order to send encrypted messages or verify digital signatures. In addition they can be informed with the latest certificate revocation information by downloading a certificate revocation list (CRL) from the directory. Therefore, almost all PKIs support LDAP and no significant rearrangements should be done from their side to use an LDAP directory for managing processes in their environments.

Apart from the fact that LDAP is already supported by PKIs, it has also attractive security features. It provides simple authentication of users with a password⁶. In addition, it supports the simple authentication and security layer (SASL) [Mey97] as illustrated in [WAHM00]. Lastly, it can be combined with TLS [DA99] as described in [HMW00]. The communication over TLS guarantees privacy and integrity of the data exchanged between an LDAP directory and a client. The password authentication can be used in combination with TLS to avoid that the password is transmitted in clear. For a thorough study on the security of directories see [Cha00].

3.2 Providing PoP with LDAP

We present the scheme for providing PoP for encryption keys using an LDAP directory.

In this scheme the CA is accepting a certification request for encryption keys. At this point the CA assumes that the private key exists and issues the certificate. After that, it creates an entry on the LDAP directory as usual. One difference is that the certificate is encrypted with the public key it contains and stored in the attribute *encryptedUserCertificate* (see Appendix A). Secondly, the CA randomly chooses an LDAP user password. Its hashed value is placed in the entry’s attribute *userPassword* for the simple authentication procedure of the LDAP. It also encrypts the password using the entity’s public key and puts it in the special attribute *encryptedUserPassword*. At this point even the user itself does not know his password, since only its hash value and an encrypted version is found on the directory.

We have introduced a new object class called *pkiUserManagement* (see Appendix A) which can hold the above described attributes. Figure 1 shows an example of such an LDAP entry along with its attributes.

The end-entity, that owns the corresponding private key, does the following: First, it downloads the encrypted password and certificate and decrypts them with its private key. Now it can use the password to bind to the directory as an authenticated user and write the decrypted certificate in its entry. With this the PoP is completed and, simultaneously, the certificate is activated enabling other entities to use it. On the contrary, for an end-entity that does not own the private

⁶ When used alone it does not really provide significant security since the password travels in clear.

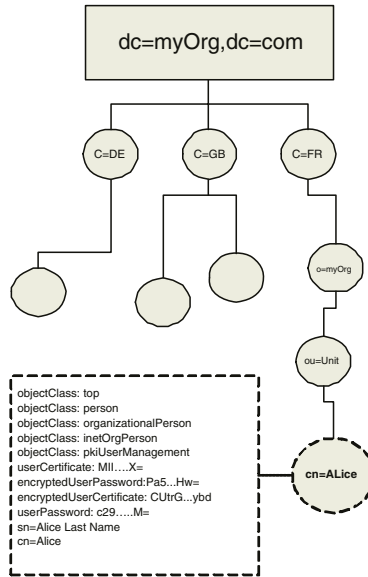


Fig. 1. End-user entry in an LDAP server.

key, both the certificate and the password will remain encrypted in the directory and cannot be used. Clearly, the communication between the authenticated user and the LDAP server must be secured with SASL or TLS to avoid transmitting the password in clear.

One step that the CA must perform is to force the LDAP directory, by proper access control lists (ACLs), to accept write requests only for authenticated users and only for the *userCertificate* attribute of their own entry. This enforces that exclusively authenticated users can place their certificate only on their own entry in the directory. In addition, these ACLs must be configured in such a way that the user password (although hashed) is not visible to other users. This will eliminate the possibility of dictionary attacks on the password performed from any end-user.

An optional variant of this scheme, which can link both identity and PoP information more tightly together, is the following: The encrypted password (when decrypted) is only the half of the real password⁷ used by the user to authenticate to the LDAP directory. The other half of the password is provided to the user during registration. This half works as a shared-secret between the CA and the end-user. The user, in order to authenticate to the directory, has to combine the two passwords.

A third variant is to have the shared-secret function as the (complete) password for the user to authenticate to the directory. PoP will follow with decryption

⁷ Which is found hashed to the directory.

	CRMF	CMP	CMC	Proposed Solution
EKS	+	+	+	+
DM	+	+	-	-
PKR	-	-	+	+
SRT	-	-	-	+

Table 1. Comparison Table.

of the certificate without having to decrypt the password too. Nevertheless, in this case the ability to authenticate to the directory is disconnected from PoP.

CAs can realise this scheme differently based on their policies. For example, if after three days the certificate is still encrypted in the directory, the user is deleted from the directory leaving him no ability to use this certificate. Alternatively, after decryption of the certificate the user password is deleted in order for the user to be unable to authenticate to the directory.

3.3 Methods Overview

We discuss the characteristics and properties of the certification request messages presented in Section 2 and our proposed LDAP based solution. PKCS#10 and XKMS do not support PoP for encryption keys (Encryption Key Support, EKS) and we will not discuss them further in this section. CMC and the LDAP based proposal specify only one possible method to provide PoP, while CRMF and CMP offer three different ones (Different Methods, DM). Nevertheless, the last ones propose revealing the private key (Private Key Revealing, PKR) as a solution. Furthermore, CMC requires more than a single-round trip where all other methods do not. But CMP and CRMF require that a confirmation message should be sent to the CA. Our proposed method does not require those (Single Round Trip, SRT). In table 1 we can see an overview of these characteristics.

3.4 Secure Delivery of Software PSEs with LDAP

A variation of the above scheme can be used for delivering software PSEs (usually in the PKCS#12 format). In this scenario the PKCS#12 structure is placed in the *userPKCS12* [Smi00] attribute of the LDAP entry. The connections done to the directory must be secured with TLS. If a user can authenticate to the directory, then he has the ability to download its own PSE. But in order to authenticate, he needs a password provided during registration. Therefore only the intended user can access his PSE.

To enforce this, the CA has to choose special ACLs. Only the authenticated users must have read access to the *userPKCS12* attribute of their own entry, while all others should have no privileges at all. Downloading the PSE is performed with integrity and confidentiality due to the TLS connection.

We have implemented both schemes to provide a proof of concept. We have used the OpenLDAP [OpenLDAP] directory since it gives the possibility to

create flexible ACLs and supports TLS and SASL. We have used JNDI [JNDI] at the CA and client side for the LDAP interfaces.

3.5 Usability Issues

We believe that these two basic applications for user management with LDAP enhances the usability of the PKI. In the PoP case it is easier for a human user to decrypt a value than to provide his private key or to engage himself in challenge-response procedures. Decrypting data like the certificate or the password is the intended usage of the decryption key and is already implemented in common client software. Furthermore, most client software comes with LDAP support to be able to fetch the certificates of others. These features can be easily extended and combined to automatically fetch the encrypted data from the directory, decrypt it and store the certificate back. Particularly, the last action can be totally transparent to the end-entity.

Also, fetching a PSE from the directory is essentially the same as looking up certificates of others. Thus, no extra development needs to be done for the client software. The usability is increased by the fact, that the client can transparently download the PKCS#12 file and integrate the contained private key at the client side⁸. Instead, if the user has received his PSE by e-mail or floppy disc, he would have to install it somehow manually. Furthermore, only the LDAP connection has to be configured at the client and nothing else. Depending on the CA's policy, also the following strategy may be chosen: The software PSE may additionally remain on the directory to serve as backup. Moreover, it can be available from everywhere. For this, client software could be configured to download and use the PSE on demand without mandatory storing it locally.

4 Conclusion

We have presented two schemes for providing proof-of-possession and secure delivery of personal security environments. Both schemes use an LDAP directory which is already integrated in most PKIs, it is ideal for user management, it has sufficient security features and many existent clients have already LDAP interfaces which can be extended. We plan to integrate our proposed solutions into an upcoming PKI installation at the University of Darmstadt. Future work will concentrate on extending the use of LDAP in PKIs for other administration and certificate management tasks like certificate revocation and certificate renewal.

References

- [AF99] C. Adams, S. Farrell, *Internet X.509 Public Key Infrastructure Certificate Management Protocols*, Request for Comments 2510, March 1999.

⁸ Probably asking for an extra password for its internal database.

- [ANL03] N. Asokan, V. Niemi, P. Laitinen, *On the Usefulness of Proof-of-Possession*, Proceedings of the 2nd Annual PKI Research Workshop, pp. 122-127, Gaithersburg MD, USA, April 2003.
- [Cha00] D. W. Chadwick, *Secure Directories*, Proceedings of the NATO Advanced Networking Workshop on Advanced Security Technologies in Networking, June, 2000.
- [DA99] T. Dierks, C. Allen, *The TLS Protocol Version 1.0*, Request for Comments 2246, January 1999.
- [DH76] W. Diffie, M. E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. IT-22, No. 6, pp. 644-654, November 1976.
- [HM02] J. Hodges, R. Morgan, *Lightweight Directory Access Protocol (v3): Technical Specification*, Request for Comments 3377, September 2002.
- [HMW00] J. Hodges, R. Morgan, M. Wahl, *Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security*, Request for Comments 2830, May 2000.
- [HPFS02] R. Housley, W. Polk, W. Ford, D. Solo, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Request for Comments 3280, April 2002.
- [JNDI] *Java Naming and Directory Interface*, <http://java.sun.com/products/jndi/> (8 Feb. 2004).
- [Mey97] J. Myers, *Simple Authentication and Security Layer (SASL)*, Request for Comments 2222, October 1997.
- [MASK99] M. Myers, C. Adams, D. Solo, D. Kemp, *Internet X.509 Certificate Request Message Format*, Request for Comments 2511, March 1999.
- [MLSW00] M. Myers, X. Liu, J. Schaad, J. Weinstein, *Certificate Management Messages over CMS*, Request for Comments 2797, April 2000.
- [OpenLDAP] *OpenLDAP Project*, <http://www.openldap.org> (7 Feb. 2004)
- [PKCS10] RSA Laboratories, *PKCS#10 v1.7: Certification Request Syntax Standard*, May 2000.
- [PKCS12] RSA Laboratories, *PKCS#12 v1.0: Personal Information Exchange Syntax*, June 1999.
- [Smi00] M. Smith, *Definition of the inetOrgPerson LDAP Object Class*, Request for Comments 2798, April 2000.
- [WAHM00] M. Wahl, H. Alvestrand, J. Hodges, R. Morgan, *Authentication Methods for LDAP*, Request for Comments 2829, May 2000.
- [XKMS] *XML Key Management Specification (XKMS)*, <http://www.w3.org/TR/2001/NOTE-xkms-20010330/> (8 Feb. 2004).
- [X.500] ITU-T Recommendation X.500, *Information Technology - Open Systems Interconnection - The Directory: Overview of Concepts, Models and Service*, Februar 2001.
- [X.509] ITU-T Recommendation X.509, *Information Technology - Open Systems Interconnection - The Directory: Public-key and Attribute Certificate Frameworks*, March 2000.

Appendix A

Object Classes

```
( 1.3.6.1.4.1.8301.3.2.2.1.6 NAME 'pkiUserManagement' SUP top
  AUXILIARY MAY ( userEncryptedPassword $ userEncryptedCertificate ) )
```

Attributes

```
( 1.3.6.1.4.1.8301.3.2.2.1.7 NAME 'userEncryptedPassword'  
EQUALITY octetStringMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 SINGLE-VALUE )
```

```
( 1.3.6.1.4.1.8301.3.2.2.1.8 NAME 'userEncryptedCertificate'  
EQUALITY octetStringMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 SINGLE-VALUE )
```

Recursive Certificate Structures for X.509 Systems

Selwyn Russell*

Information Security Research Centre
Queensland University of Technology
Brisbane, Australia
S.Russell@qut.edu.au

Abstract. Path discovery and path building in large scale certificate based infrastructures can be difficult and a hindrance to secure electronic communications. The recursive certificate structure can eliminate or greatly alleviate these problems. This paper presents in ASN.1-like syntax a description of how the recursive certificate structure could be employed in an X.509 centric system.

Keywords: recursive certificate, X.509, path building, path discovery, PKI, PMI

1 Introduction

Two related potentially serious problems in communications between heterogeneous large scale Public Key Infrastructures (PKIs) are path discovery and path validation of chains of certificates [1,2,3]. This research addresses both of these problems, particularly path discovery.

In X.509 systems, it is assumed that each participant has one or more certificates which contain certified information about the participant. In this paper, we represent a certificate in the form $S_{Alice}(C_{Bob})$ meaning that the certified content about Bob has been signed by the issuer of the certificate, Alice. When there are a number of issuers of certificates in succession, a chain of certificates is formed, such as $S_1(C_1), S_1(C_2), S_2(C_3), S_3(C_4)$, where $S_1(C_1)$ is the self signed certificate of the root Certification Authority (CA), the first entity in this sequence of four entities, 1 through 4. With secure message systems the addressee may receive a set of certificates involving one or more chains of certificates from the sender of the message back to the root issuer(s). Path discovery is arranging individual certificates into a chain, and path validation is verifying that all components in a discovered chain comply with all security constraints and requirements [1]. Although trivial for the case of a home shopper using a browser with SSL to communicate with a merchant's server, path discovery and reconstruction becomes a significant problem with interoperation between large scale Public Key

* This work was partially funded by the Telecommunications Advancement Organisation of Japan

Infrastructures involving multiple authorities, as in global trade. Complications arise when two or more chains for an end entity exist, and when there is cross-certification or bridging. The IETF PKIX group has issued a 74 page draft RFC [2] (third version) on the path building problem, so that by “following the guidance and recommendations defined in this document, an application developer *is more likely* to develop a robust X.509 certificate enabled application that can build valid certification paths across a wide range of PKI environments”. Note the words “is more likely to” rather than “can” or “will”.

A major cause of the difficulties lies with the traditional practice of issuing separate certificates for entities. We have offered a suggestion [4,5] for Certification Authorities to issue a single data structure containing multiple certificates, being the correct path components up to that point, called a “Recursive Certificate” structure because of the algorithm for its construction. This work was generic and applicable to many types of certificates, and no information on any specific implementation was given. Here we show how the generic recursive principles can be applied to X.509 certificates.

This paper addresses PKIs, CAs, and public key certificates in particular, but the principles apply similarly to X.509 PMIs, AAs, and attribute certificates.

1.1 Contributions and Structure of the Paper

In section 2, we briefly review the previous work on the generic recursive certificate structure and its limitations. We provide a specification in an ASN.1-like style for that structure in section 3. The aim of this paper is to publish guidelines and a framework for implementation, rather than a detailed syntactically perfect specification, for which there is insufficient space anyway. We point out the problems with this implementation and suggest, in section 4, a rearrangement of content and modified algorithm. In section 5, we provide another structure, still in conformity with the recursive certificate principle of a single data structure which allows an existing X.509 program to read at least the content of the most recent entity in a certificate chain, usually the sender of a message in the context of the path validation problem. In section 6, we extend the basic principle of the structure in section 5 to a non-recursive version using the X.509 extension mechanism. A summary is provided in section 7.

2 Review of Original Generic Recursive Certificate Structure

A recursive certificate assists a validator of a certificate associated with a hierarchy or mesh [2] of CAs. In the current situation, a validator of a certificate of an end entity receives a set of certificates purporting to be components of a path terminating with the end entity, one certificate for each CA in the path and one certificate for the end entity. In the situation of an end entity having a recursive certificate which began at the root CA, only one certificate is received by the validator: the recursive certificate of the end entity. The certificates of

the individual CAs are not sent; their information is in the recursive certificate of the end entity.

2.1 Outline of System and Benefits

In the generic system, using the terminology shown in the Introduction, the n th CA entity in a chain issues a recursive certificate to the $n + 1$ th entity as a single data structure described as $S_n(RC_n, C_{n+1})$ where RC_n is the recursive certificate of the n th CA. Showing all of the components, this is

$$S_n(S_{n-1}(\dots(S_2(S_1(S_1(C_1), C_2), C_3), \dots C_n), C_{n+1}))$$

Benefits [5,3] include the elimination of the path discovery problem and potentially reduced validation effort. Another benefit is the potential elimination of validation of certificates earlier in the chain than a designated trusted CA. With the recursive certificate structure, the signature of the issuing CA shows the inclusion of higher level certificates was by that CA (not an attacker), starting with the trusted root CA. If the receiver considers a particular CA to act trustworthily, then there is some degree of assurance in that CA's signature that the CA complied with policies and constraints in issuing the certificate. A validator which trusts the first n CAs will benefit from the signatures of those CAs in a recursive certificate. For example, suppose a fictitious company, XYZ Corp in an Asian country with a government root CA (e.g. [6,7,8]) has a corporate CA which issues to members of its staff. The government root CA is only a reference point and provides certificates to authorised second level CAs which interact with corporate CAs, other CAs and the public. Partner firms in Europe which receive messages from officers of XYZ would normally process the entire chain from the officer through the corporate CA and authorized higher level CAs terminating at the government root CA, one certificate at a time. With a recursive structure, if the partner firm trusted the corporate CA, it would not need to validate separately the signatures of the higher level CAs, because the recursive structure is, inter alia, a statement by the issuing CA that it is acting in accordance with policies and limitations and that it has validated its higher level certificates at the time of issue to a subordinate subject. If the validator trusts the corporate CA, it need process no further. If the validator does not trust the corporate CA, it goes ahead and extracts from the recursive certificate the individual certificates of higher level CAs one at a time and processes them, as is the current situation but without the burden of path discovery. With separate certificates, the validator has no choice but to process each and every one, for there is a risk that the some of the supplied higher certificates have been substituted by an attacker and are not the ones used by the CA. As well as providing the correct path, the recursive certificate structure permits assurance that no substitutions by an attacker have occurred.

Wireless users also are likely to benefit from recursive certificates. Wireless devices are typically computationally constrained and would take longer to process a chain of certificates than a desktop computer. Use of a recursive certificate structure can potentially speed processing for m-commerce, reducing latency, network traffic and costs.

3 Adaptation to the X.509 Environment

Starting in this section we take the published generic recursive certificate principles and suggest a way to apply them in an X.509 world. We begin with a normal X.509 description in ASN.1 and develop ASN.1-like syntaxes for recursive certificates to four levels, then move to a more compact specification based on CHOICE in 3.2, with comments in 3.3. Omission of several top level certificates is discussed in 3.4. A better idea than excluding a certificate entirely may be to replace it with a unique reference to the original, and is discussed in 3.5. The likely behaviour of a current X.509 processor when confronted with a recursive certificate of the model in this section is the topic of 3.6. We then move into the next section and consider an alternative ASN.1-like specification providing contents for subjects in a different sequence.

3.1 Extending the X.509 Specification to the Recursive Structure

The X.509 certificate specification has evolved through three versions and is well known. Descriptions of the specification frequently begin with something like the following [9]:

```
X.509Certificate ::= SEQUENCE
{
  tbsCertificate      TBSCertificate,
  signatureAlgorithm  AlgorithmIdentifier,
  signatureValue      BIT STRING
}
```

The first certificate in a recursive structure is the X.509 certificate of the top level CA. The certificate issued by the root CA to the level 2 entity would be

```
Level2Certificate ::= SEQUENCE
{
  rootCaCert                X.509Certificate,
  tbsCertificateLevel2Entity TBSCertificate,
  signatureAlgorithm         AlgorithmIdentifier,
  signatureValueByRootCA     Bit String
}
```

Proceeding down the path (a hierarchy or a mesh [1,2]), the level 2 entity issues a recursive certificate to the level 3 entity and the structure becomes: the recursive certificate for the issuer (level 2 entity) followed by the TBSC for the subject (level 3 entity) and completed with the signature of the issuer.

```
Level3Certificate ::= SEQUENCE
{
  level2EntityCert          Level2Certificate,
  tbsCertificateLevel3Entity TBSCertificate,
```

```

signatureAlgorithm      AlgorithmIdentifier,
signatureValueByLevel2Entity Bit String
}

```

which is

```

Level3Certificate ::= SEQUENCE
{
  SEQUENCE
  {
    rootCaCert          X.509Certificate,
    tbsCertificateLevel2Entity TBSCertificate,
    signatureAlgorithm   AlgorithmIdentifier,
    signatureValueByRootCA Bit String
  }
  tbsCertificateLevel3Entity TBSCertificate,
  signatureAlgorithm       AlgorithmIdentifier,
  signatureValueByLevel2Entity Bit String
}

```

We will proceed no further with the details for deeper levels, which will be left as exercises to the interested reader.

3.2 A More Compact Specification

As the hierarchy depth increases, the nested depth increases with it, and the size of the ASN.1-like definition grows. We can represent a recursive certificate more briefly as either a standard X.509 certificate or a recursive certificate, the former being true for the first in the chain, and the latter applying to all subsequent certificates.

```

RecursiveX.509Certificate ::= CHOICE
{
  standardCert X.509Certificate,
  recCert RecursiveX.509Certificate
}

```

which can be written in more detail as

```

RecursiveX.509Certificate ::= CHOICE
{
  SEQUENCE
  {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm   AlgorithmIdentifier,
    signatureValue       Bit String
  },
}

```

```

SEQUENCE
{
  recCertOfIssuer      RecursiveX.509Certificate,
  tbsCertificateOfSubject TBSCertificate,
  signatureAlgorithm    AlgorithmIdentifier,
  signatureValue        Bit String
}
}

```

3.3 Comments on Processing

The certificate content of the top of the hierarchy appears first when read sequentially, with later contents being added at the end, so the content for the most recent subject will be the final one read. A receiver will most likely be concerned with the most recent subject, presumably the sender of the signed message, and the receiver will have to process content for higher entities before reading the most recent subject's details. On the other hand, those wishing to perform their own complete validation of every certificate in the chain starting from the root will find this ordering ideal.

How does a receiver know which is being sent to it? In the case of the *X.509Certificate*, the certificate is constructed non-recursively and the DER encoding [10] which is received by the validator begins with "SEQUENCE length INTEGER length ..." (the *INTEGER* component corresponds to the *Version*), while every certificate constructed recursively will be encoded as "SEQUENCE length SEQUENCE length ...", with at least two "SEQUENCE" at the beginning before an "INTEGER". The receiver's recursive certificate decoding program will recognize a recursively constructed certificate from the initial octets. A program developed to handle recursive X.509 certificates will handle standard X.509 certificates also, however a program not aware of recursive certificates would report an error in the input structure.

3.4 Starting the Recursive Certificate at a Lower CA

In the more general case where the upper level CAs and their certificates are well known, e.g. in an Asian government sponsored PKI, an issuer may wish to omit several of the uppermost entities' certificates and begin the recursive certificate with a lower CA.

We could have a situation where the certificates of the first four entities ($S_1(C_1)$, $S_1(C_2)$, $S_2(C_3)$, $S_3(C_4)$) are well known and do not appear in a recursive certificate such as $S_6(S_5(S_4(C_5), C_6), C_7)$.

The reader will see that the certificate for the first entity in the recursive certificate will be a normal X.509 certificate, and the certificates for lower entities will contain that X.509 certificate plus other details as outlined above. The former option of the CHOICE definition applies to the first CA in the recursive certificate, and the other to later CAs and the final subject.

If a validator seeks to build a complete path, it will be necessary to locate and verify the upper non-recursive certificates, which may be complicated by CAs having multiple public key pairs for different purposes, as is often the case nowadays. (PKIX RFC 3280 [9] allows for nine standard types of key usage, and hence up to nine different keys if only one is used for each purpose, so a large CA could have several dozen different public keys.) With a little more syntax, we can assist with the location of the appropriate missing certificates, as we show in the next sub-section.

3.5 Including a Certificate Reference

Leaving out well known CA certificate saves space in the recursive certificate of a later CA and assists transmission latency but is not recommended, particularly for public cellular wireless, because the validator has to process as individual certificates the upper level certificates which have been omitted. A minor complication is the fact that the unspecified well known CAs probably have more than one certificate each and the validator may have to expend resources to determine which ones were used to form the chain in question. A compromise would be to include an identifying reference in the recursive certificate, e.g. issuer name, issuer's serial number and fingerprint, which we now explore.

Adding a reference R_1 to an omitted root CA certificate gives us

$$S_n(S_{n-1}(\dots(S_2(S_1(R_1, C_2), C_3), \dots C_n), C_{n+1}))$$

For the X.509 environment, we can introduce a CertID component which contains a unique identification which can be tested by a validator holding a “well known” certificate to determine if the certificate in the chain but which was omitted from the recursive certificate is the same as possessed by the validator. For example,

```
CertID ::= SEQUENCE
{
    issuerName Name,
    serialNum CertificateSerialNumber,
    fingerPrint BIT STRING
}
```

Using the certificate reference to the specific certificate of the trusted root CA, the level 2 example of 3.1 can be amended to

```
Level2Certificate ::= SEQUENCE
{
    rootCaCertId CertID,
    tbsCertificateLevel2Entity TBSCertificate,
    signatureAlgorithm AlgorithmIdentifier,
    signatureValueByRootCA Bit String
}
```

which is larger than when the root CA certificate is omitted entirely but smaller than when it is included as in 3.1.

Using a device like this allows us to save some space while providing more assurance to a validator.

3.6 Backward Compatibility of This Specification

As in 3.3, a new program will handle recursive and non-recursive structures, but a program which does not understand recursive certificates would process only the former CHOICE option and will fail when it encounters the latter option.

In the case where the first n CAs are trusted, the validator wants fast access to the contents of the later certificates but with the above structure it will have to read the already known trusted content first. To change this situation, we amend the structure in the next section.

4 An Alternative Structure

In this section we will present an outline of the alternative structure, and a specification in an ASN.1-like syntax. An example will show the modified forms of the certificates for levels 2, 3 and 4 of the previous section. We discuss processing and backward compatibility.

4.1 Outline of the Alternative Recursive Structure

For this variation, the construction algorithm of 2.1 is changed so that a CA prepends, rather than appends, the details of the subject to the CA's recursive certificate, and then signs the combination, as $S_n(C_{n+1}, RC_n)$.

The major difference between this format and the earlier one in 3.1 is that the details of more recent subjects are towards the front. The content for the highest entity in the chain will be at the end.

4.2 ASN.1-like Specification

We can retain the overall principles of chain declaration and recursivity but alter the details by devising an alternative structure which is still described as

```
RecursiveX.509Certificate ::= CHOICE
{
    standardCert  X.509Certificate,
    recCert      RecursiveX.509Certificate
}
```

by defining

```
RecursiveX.509Certificate ::= CHOICE
{
  SEQUENCE
  {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      Bit String
  },
  SEQUENCE
  {
    tbsCertificateOfSubject TBSCertificate,
    recCertOfIssuer
      RecursiveX.509Certificate,
    signatureAlgorithm      AlgorithmIdentifier,
    signatureValue          Bit String
  }
}
```

4.3 Processing Algorithms

To generate a recursive certificate for a client, a CA creates the certificate content information (the *TBSC* component) pertaining to the client, appends its own recursive certificate, and adds a digital signature. For the purposes of this procedure, we regard a self signed certificate of a root CA to be a recursive certificate.

If a receiver seeks to validate all of the certificates in the chain, it can extract them out of the recursive structure, giving a chain of separate certificates which can be processed in the normal manner.

4.4 Backward Compatibility

As with the original structure in 3, a program which is able to process only separate X.509 certificates will not be able to process any part of this X.509 structure. While this is not a problem for new applications, we considered a new structure for a recursive certificate path which would allow a legacy X.509 reader to extract information about the most recent entity in the recursive certificate, which is probably the sender of the message which is being validated. This new structure is reported in the next section.

5 A Backward Compatible Alternative

The previous two structures involved no change to the *tbsCertificate* cluster in the X.509 certificate but the encoding for the added nested *SEQUENCE*

items at the beginning meant that a current X.509 program would not accept these structures. In this section we will describe a construction option which will allow a current X.509 program to at least read and process the details for the most recent subject, even though it would not be able to understand the content for higher level entities. We achieve this goal by modifying the *tbsCertificate* portion itself, with no other changes to the standard X.509 structure, as outlined below.

5.1 Outline of the Principle

Version 3 of X.509 introduced mechanisms for certificate issuers to create new values in certificates, called “extensions”, provided they met a certain very broad format. Programs which process version 3 X.509 certificates are able to recognize that an extension is an extension, even though they may not be programmed to be able to correctly process the values in the extension.

For the alternative recursive certificate of this section, the earlier certificate information is included in a new type of extension, inside the certificate contents rather than externally as in sections 3 and 4. The extension will contain the recursive certificate of the issuer, as received from its issuing parent CA. An implementation is:

```
IssuerCertsExtn ::= SEQUENCE
{
    issCertsExtnID OBJECT IDENTIFIER,
    critical          BOOLEAN FALSE,
    issCertVal        OCTET STRING
}
```

The first CA in the recursive certificate would use an extension with values as shown below, i.e. it includes its own self signed X.509 certificate in the extension in the certificate for the next level entity.

```
SEQUENCE
{
    issCertsExtnID OBJECT IDENTIFIER,
    critical          BOOLEAN FALSE,
    rootCaCert        X.509Certificate
}
```

Later CAs build up recursively.

5.2 Processing Algorithms

For an issuer, the first step is to make the normal *tbsCertificate* content for the next subject, then make the *IssuerCertsExtn* extension for this certificate, which is added to the output from the first step, and the total data block signed to form the certificate for the subject.

As earlier in subsection 3.1, the beginner of the recursive certificate data structure can include its normal X.509 certificate in the recursive certificate data structure (or recursive certificate if desired and if available); others must include their recursive certificate as delivered to them by their parent CA.

For a validator which is unaware of the non-critical recursive extension, the details of the subject will be obtained and the unknown extension will be ignored. Other certificates, if needed, will have to be obtained in the normal way. For a validator which is aware of the recursive content, those details can be extracted offline and without further network activity (other than any online revocation checks).

The next section explains a further proposal which would allow inclusion of CA certificates in a non-recursive manner in a similar but different extension.

6 A Simpler Non-recursive Alternative

In this section, we propose using a non-recursive *SEQUENCE* of standard certificates, allowing a standard X.509 processor to read the content for the most recent subject, and, with some modifications to extract the included certificates, process the chain of certificates in the normal way.

We first outline the principle, give a more formal specification, and discuss processing algorithms, size reduction and disadvantages.

6.1 Outline of the Principle

As in 5.1, we modify the standard *TBSCertificate* with a new extension that contains certificate information. In this alternative, we are avoiding recursion processes, so we use the equivalent of an array of non-recursive certificates and these certificates, after extraction, are accepted by all current X.509 processors. Some modifications would be needed to current X.509 processors to extract the certificates from the proposed extension. Removing the recursive structure introduces other complications which will be explored below.

6.2 ASN.1-like Specification

Here we use *IssuerCertExtn* rather than the *IssuerCertsExtn* of 5.1 because we will be working with individual certificates rather than a recursive data structure of a set.

```
IssuerCertExtn ::= SEQUENCE
{
    issCertExtnID OBJECT IDENTIFIER,
    critical      BOOLEAN FALSE,
    issCertVal    OCTET STRING
}
```

For the first entity in the path, the root CA, this would be

```
SEQUENCE
{
  issCertExtnID OBJECT IDENTIFIER,
  critical      BOOLEAN FALSE,
  rootCaCert    X.509Certificate
}
```

6.3 Processing Algorithms

Processing is more complicated for the issuer than in the previous proposal in 5.2.

When a CA issues a certificate similar to that above in 6.2, it should include its simple X.509 certificate in the list of certificates in the path. However, if a CA receives from its issuer only a certificate similar to that above and no simple X.509 certificate, then it does not have a simple X.509 certificate to include. If it includes its non-recursive certificate of 6.2 in the certificate it issues to a subject, its certificate entry in the list will contain the certificates of CAs on the full prior path. If every CA in a path of n CAs issues in this manner, not having a simple X.509 certificate of its own, then in the end entity's certificate there will be n copies of the root CA's certificate, $n - 1$ copies of the level 1 CA's certificate, etc. To avoid this unacceptable growth of size, we consider two options.

One option is to have every CA issue to a subject which is a CA, both a simple X.509 certificate and an extended certificate as in 6.2. A CA is then able to extract the simple certificates of the higher level CAs from its extended certificate, and has its own simple X.509 certificate which it can include in the extended certificates it issues to its subjects. The procedure to create two similar signed data blocks is fairly simple, but will cost another digital signature, and involve increased ongoing certificate management expenses.

The other option is to have CAs issue only simple X.509 certificates to other CAs, and have a CA issue only extended certificates to end entities. This option requires the CA to know exactly the path of the higher entities by other means, which it probably does and often the path will be invariant. That list will be included in extended certificates issued to its end entity subjects. A certificate validator is interested in the path for an end entity which has sent secure messages, and an end entity will have a suitable extended certificate with this option. There are no changes to issuance procedures for high level CAs; only the CAs which issue to end entities need to create extended certificates.

To create the certificate, the CA forms the normal content, then forms the extension list of earlier CAs from its simple X.509 certificate and those of the other CAs, and signs the combined block.

For the validator receiving this style of certificate, extraction of the simple X.509 certificates is fairly easy once the extended list is recognized. Normal X.509 path validation can then proceed.

6.4 Reducing the Size

As in 3.5, we could simply replace a complete certificate with a unique reference to it, saving space but requiring a validator seeking it either to have the certificate on hand or to fetch it from some network location which must be somehow specified or inferred, and requiring online network access by the validator. For a wireless client, the smaller size is attractive for the initial download but the subsequent additional network accessing will increase the processing latency and usually result in higher call charges.

6.5 Disadvantages of the Non-recursive Form

In this format, the certificates of the higher level entities can be extracted more simply, by reading them sequentially. However the certificate of a higher level CA could have been included without its knowledge, and, without the signature of the CA or alleged CA for verification, lacks the assurance provided in the recursive structure. A diligent validator will need to verify policies and constraints of all certificates, but will still benefit from not having to conduct path discovery and path building.

There is also the problem for a CA of what to issue to other CAs, as discussed in 6.3.

7 Summary

We have taken a generic description of recursive certificates and adapted it to the X.509 environment in four ways. In section 3, we used the earlier generic proposal of appending the content for the next subject to the issuer's recursive certificate, and in the following section we modified this so a validator could access the end entity's certificate content first rather than last. We explained that both of these structures could not be understood by unmodified current X.509 processors.

We proposed in section 5 a recursive certificate more similar to current usage by adding a nested set of recursive certificates as a single extension in any particular certificate, permitting a current X.509 processor to at least obtain the details of the most recent subject, usually the sender of a secure message.

Moving to a more legacy compatible form, we proposed in section 6 a different extension to X.509 as an array of X.509 certificates, losing some of the important benefits of recursive certificates but still retaining the benefits of inherent path discovery and path building. Issuance is complicated by the requirement that a CA has a simple X.509 certificate to include in the extended certificate, raising an unattractive possibility that a CA might need to issue both a simple and an extended certificate to subjects which are CAs.

Of course, certificate references could be substituted for actual certificates in many cases, as discussed in 3.5.

References

1. Russ Housley and Tim Polk, *Planning for PKI*, Wiley, 2001.
2. Michael Cooper, Yuriy Dzambasow, P Hesse, S Joseph, and R Nicholas, *Internet X.509 Public Key Infrastructure: Certification Path Building*, Dec. 2003, draft-ietf-pkix-certpathbuild-03.txt.
3. Selwyn Russell, "A Different Approach to Certificate Path Discovery and Validation," in *Proceedings of the IASTED International Conference on Communications, Internet and Information Technology*. Nov. 2003, IASTED, Held at Scottsdale, Arizona.
4. Selwyn Russell, "Recursive Certificates: a New Paradigm for PKI Certificates," in *Proceedings of Second International Workshop for Asian Public Key Infrastructure*. 30 October – 1 November 2002, pp. 14 – 20, Chinese Cryptology and Information Security Association, Taiwan.
5. Selwyn Russell, "Theory and Benefits of Recursive Certificate Structures," *International Journal of Information Security*, 2003.
6. Heesun Kim, Yeongsu Cho, Seunghun Jin, and Kyoil Chung, "Current Status and Trends of PKI in Korea," In Kim [11], pp. 1 – 21.
7. Satoru Tezuka, "Trend of Japanese PKI and International Cross Certification," In Kim [11], pp. 22 – 31.
8. Shuo Bai, "PKI in China," In Kim [11], pp. 40 – 50.
9. R. Housley, W. Ford, W. Polk, and D. Solo, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, Apr. 2002, RFC 3280.
10. Burton S. Kaliski Jr., *A Layman's Guide to a Subset of ASN.1, BER, and DER*, RSA Data Security, Inc., Redwood City, CA, 1 November 1993, (Supersedes version of 3 June 1991).
11. Kwangjo Kim, Ed., ICU, Daejeon, Korea, 19–20 October 2001. International Research Center for Information Security, Korea and Institute of Industrial Science, Japan.

A Probabilistic Model for Evaluating the Operational Cost of PKI-based Financial Transactions

Agapios Platis, Costas Lambrinoudakis, and Assimakis Leros

Department of Information and Communication Systems Engineering,
University of the Aegean, Karlovassi,
83200 Samos, Greece
{platis, clam, aleros}@aegean.gr

Abstract. The use of PKI in large scale environments suffers some inherent problems concerning the options to adopt for the optimal cost-centered operation of the system. In this paper a Markov based probability model has been applied and a performability indicator has been introduced for assisting the evaluation of the operational cost of the system in a decision support process. Considering the unavailability of the Certification Authority server, three different strategies have been evaluated for determining the optimal one.

1 Introduction

During the last decade the Public Key Infrastructure (PKI) has been widely used for the provision of security services, especially in application domains like e-commerce, financial transactions, e-health etc. A central role of a PKI is that of the Certification Authority (CA) that mainly deals with issuing key pairs (a private and a public key) for customers[14], or/and simply register a public key supplied by the registered entity. The private key must remain secret, under the control of its owner, while the public key must become available to anyone wishing to have some type of transactions with the owner of the private key. At this point the concept of a *digital certificate* is introduced, linking the public key of a customer with her/his identity. This linkage is certified and digitally signed by the CA and, therefore, trusted by any two parties utilizing the PKI for performing a transaction. However, any digital certificate has an expiration date and frequently unexpired certificates must, for some unexpected reason (the private key has been compromised, user credentials have changed etc), be invalidated (*revoked*)[5, 6]. In either case the certificate must not be used. It is therefore clear that a mechanism supporting an entity to confirm the validity of some other entity's certificate must exist.

Periodically-issued *Certificate Revocation Lists* (CRLs) are one common approach to revoking certificates; each such list specifies what unexpired certificates have been revoked, and when the next CRL will be issued. The issuing CA signs the CRL and someone wishing to check the validity of a certificate must download the "most recent" CRL from the CA. However, the "recency" of the CRL is a parameter that should reflect the requirements of each specific customer, implying that someone could be satisfied with weekly updated CRLs while someone else could require at most day-old evidence[15].

In this paper we present a probabilistic model that can be employed for evaluating the “cost” of different strategies as far as the certificate validation checks are concerned. Specifically, depending on the transaction characteristics, an entity may choose to trust the provided certificate and proceed with the transaction without confirming its validity –in this case the specific entity accepts the risk of a *security incident* to happen due to invalidated or/and expired certificates. Alternatively, for a different transaction the same entity may set as a prerequisite for carrying out the transaction the confirmation of the certificate’s validity through the CA’s CRL. However, the desired confirmation may not be possible due to CRL (CA) unavailability, a term used in this paper for describing one or more of the following cases:

- The CA server (and thus the CRL) is not accessible.
- The CRL is not accessible / available.
- The “recency” of the CRL does not fulfill the user requirements.

It is therefore not possible to proceed with the transaction (*failed transaction*) if the CRL-CA is unavailable (irrespective of the reason causing the unavailability) for a period exceeding a specified threshold.

In order to evaluate the different strategies, a Markov based Performability model has been used. Performability modeling or Markov Reward Models (MRM) have been initiated by Beaudry [1] and Meyer [8] and successfully used for evaluating the performance of gracefully degrading computer systems, cf.[2,3,4,10,17] and electrical systems, cf.[12], but also for evaluating the quality of Internet services, cf.[18,19], or website reorganization, cf.[13]. The major advantage of such models is that they combine reliability and performance measures, including to a greater extent cost related measures and external environmental parameters and thus allowing a more detailed modeling. Indeed, for a high availability system, a failure during peak hours has a much greater impact on users than a failure occurring when the system is not extensively used. Where classical availability evaluations cannot differentiate these events, the performability model can perceive and evaluate each occurrence individually. In the same way a classic Markov model will give important information about the probabilities of a security incident and a failed transaction. A security incident is, of course, a highly undesirable event, normally causing much more serious consequences than those caused by a failed transaction. On the other hand, a large amount of failed transactions may have a higher cumulative cost than a single or a few security incidents. By utilizing a performability model the above-mentioned parameters can be taken into account.

2 Scenario Description

This section describes the PKI architecture and the operational scenarios that have been chosen for modeling (Section 3). It is evident that different implementations, either in terms of the certificate revocation mechanisms or/and in terms of supported functionality, would cause differentiations in the model presented in this paper. However, the scenario that has been adopted can be characterized as *representative* of a typical general-purpose PKI implementation capable of supporting a wide range of applications.

Specifically, the existence of a Certification Authority (CA) is assumed, which in addition to the task of generating and distributing key pairs to customers it is responsible for maintaining a certificate revocation list (CRL) in order to allow anyone interested to check, prior to a transaction, the validity of someone else's certificate. For the purposes of the current paper, it is assumed that the CRL can only be checked if the CA server is available. This assumption is based on the fact that even if the customers maintain local copies of the CRL, prior to a high valued financial transaction they will always request an up-to-date CRL from the CA. Based on the above, a typical operational scenario is the following:

1. A financial institution X supports on-line transactions (money orders, investments etc) for users (customers) that are registered with the specific service and have obtained a pair of valid keys from a certification authority (CA).
2. A party Y (customer) wishing to perform a financial transaction, through the institution X, submits the appropriate request, digitally signed, to X. For the digital signature X is using her/his private key.
3. The institution X, before serving the request submitted by Y, must either assume that the public key of customer Y is valid or must verify its validity through the CRL maintained by the CA. Therefore the alternative actions of X are the following:
 - a) If the amount of a transaction is below some threshold value F, X decides not to check the validity of Y's certificate and proceeds serving the transaction
 - b) If the amount of the transaction exceeds the threshold value F, X must check the validity of Y's certificate by accessing the CA's CRL list.
4. For transactions that the validity of Y's certificate was required, the following alternative paths are possible:
 - a) The CRL maintained by the CA server is accessible and thus the validity check can be performed.
 - b) The CRL is, for some reason, unavailable. If this is the case then the institution X waits for a predetermined period of time for the CRL to become available and then proceed as in 4a.
 - c) If the CRL is unavailable (case 4b) and the waiting time exceeds a threshold, the institution X cancels the transaction (*failed transaction*).
5. For transactions that the validity of Y's certificate was not checked (case 3a), there is a possibility for a *security incident* to occur.
6. Even for transactions that the validity of Y's certificate has been checked, there is a possibility for a *security incident* to occur (although with significantly smaller possibility than that in case 5).

3 PKI Markov Modeling

3.1 State Transition Diagram

The process is described as follows: transaction requests addressed to the financial institution X, arrive at a rate λ_1 which is modeled by a Poisson process, hence the inter-arrival times are exponentially distributed, β is the probability to have a transaction

with an amount exceeding the threshold value F and therefore imposing the need to check the customer's certificate, γ_1 is the unavailability probability of CA's server (and thus unavailability probability of CRL's) and in this case the system waits till the server becomes available (the server restoration rate is μ_{rest}). If the waiting time exceeds a threshold value then the transaction fails with a rate μ_2 (which is the inverse of the mean delay of a failed transaction). The service rate is μ_1 and the security incident rate is μ . A security incident may occur if no access to the revocation list has been achieved with a rate λ_3 , however a security incident can also occur even if the revocation list has been accessed but with a less important hazard rate λ_2 .

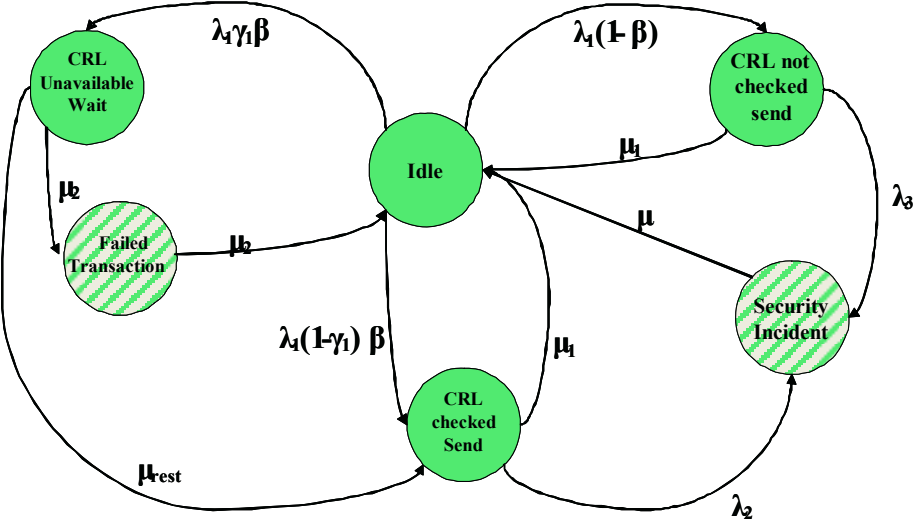


Fig. 1. State Transition Diagram.

Let X_n be the Markov Chain modeling the operational behavior of the system with state space $\{\text{Idle}, \text{CRL-U}, \text{CRL-C}, \text{CRL-NC}, \text{FT}, \text{SI}\}$ and transition rates matrix A .

Table 1. System States

Description of the System States		
Idle	Idle	System is idle, waiting for requests
CRL-U	CRL Unavailable - Wait	System is waiting for accessing the CRL
CRL-C	CRL checked - Send	The request is processed after checking the CRL
CRL-NC	CRL not checked - Send	The request is processed without prior check of the CRL
FT	Failed Transaction	The transaction has failed due to the fact that the CRL stays unavailable for a time period exceeding a threshold value
SI	Security Incident	The transaction data have been altered

Since the chain is irreducible and aperiodic, then the chain is ergodic and has a unique steady state probability distribution π .

The computation of the steady state probability distribution is obtained by solving a linear system $\pi \cdot \mathbf{A} = \mathbf{0}$ with the additional condition $\pi \cdot \mathbf{1} = 1$, where $\mathbf{1}$ is a 6-dimensional column vector containing ones and π the 6-dimensional row vector containing the steady state probabilities of the system (steady state probability distribution), cf.[9].

The resolution of the previous system gives the following results concerning the steady state probabilities of the failed transaction and the security incident states.

$$\pi_{FT} = \frac{\frac{\lambda_1 \gamma_1 \beta}{\mu_{rest} + \mu_2}}{1 + \frac{2\lambda_1 \gamma_1 \beta}{\mu_{rest} + \mu_2} + \frac{\lambda_1 \beta}{\mu_1 + \lambda_2} \left(1 - \gamma_1 + \frac{\mu_{rest} \gamma_1}{\mu_{rest} + \mu_2} \right) \left(1 + \frac{\lambda_2}{\mu} \right) + \frac{\lambda_1 (1 - \beta)}{\mu_1 + \lambda_3} \left(1 + \frac{\lambda_3}{\mu} \right)} \quad (1)$$

and

$$\pi_{SI} = \frac{\frac{\lambda_1}{\mu} \left[\frac{\lambda_2 \beta}{\mu_1 + \lambda_2} \left(1 - \gamma_1 + \frac{\mu_{rest} \gamma_1}{\mu_{rest} + \mu_2} \right) + \frac{\lambda_3 (1 - \beta)}{\mu_1 + \lambda_3} \right]}{1 + \frac{2\lambda_1 \gamma_1 \beta}{\mu_{rest} + \mu_2} + \frac{\lambda_1 \beta}{\mu_1 + \lambda_2} \left(1 - \gamma_1 + \frac{\mu_{rest} \gamma_1}{\mu_{rest} + \mu_2} \right) \left(1 + \frac{\lambda_2}{\mu} \right) + \frac{\lambda_1 (1 - \beta)}{\mu_1 + \lambda_3} \left(1 + \frac{\lambda_3}{\mu} \right)} \quad (2)$$

The above equations have been evaluated using the empirical hazard rates and probabilities listed in Table 2 below. It should be mentioned that empirical data have been used in order to demonstrate the applicability of the model.

Table 2. Hazard Rates and Probabilities of the System Parameters

Hazard Rates and Probabilities		
λ_1	Transaction request arrival rate	50 h ⁻¹
λ_2	Security incident rate given that the CRL has been checked	0,000001 h ⁻¹
λ_3	Security incident rate given that the CRL has not been checked	0,001 h ⁻¹
β	Probability of transaction exceeding threshold F	Variable 0; 1; 0,4
γ_1	CRL's unavailability probability	0,001
μ_1	Service rate	500 h ⁻¹
μ_2	1/ Mean delay of a failed transaction	100 h ⁻¹
μ_{rest}	Restoration rate for the CA-customer link	1h ⁻¹
μ	1/ Mean duration of a security incident	500 h ⁻¹

It should be stressed at this point that the probability of a security incident to occur when the validity of the customer's certificate has been checked –CRL has been accessed successfully-- (λ_2), has been assumed to be much smaller than the respective probability of a security incident when the financial transaction is performed without prior validation (λ_3). Furthermore, based on existing statistical information, the CA

server is expected to be unavailable, in average, one time every one thousand transactions (γ_1).

The calculation of the probability of *Failed Transactions* and *Security Incidents* has been performed with three different values of (β), thus simulating the operational scenarios described in section 2. Specifically:

- **Strategy 1:** $\beta=1$, validation of customer’s certificate is required prior to any transaction (CRL must be checked --- section 2, case 3b with a threshold value, F , for the amount of the transaction, set to zero).
- **Strategy 2:** $\beta=0$, all transactions are served without prior validation of customer’s certificate (CRL is not checked --- section 2, case 3a with a threshold value, F , for the amount of the transaction, set to an extremely large value).
- **Strategy 3:** $\beta=0.4$, a combination of strategies 1 and 2, expecting 40% of the requested transactions to require validation of the customer’s certificate (implying that the amount of the transaction exceeds the threshold value F), while the remaining 60% to be served without prior validation (implying that the amount of the transaction is below the threshold value F).

The calculated probabilities for Failed Transactions and Security Incidents, for each strategy, are depicted in Fig. 2 and 3 respectively.

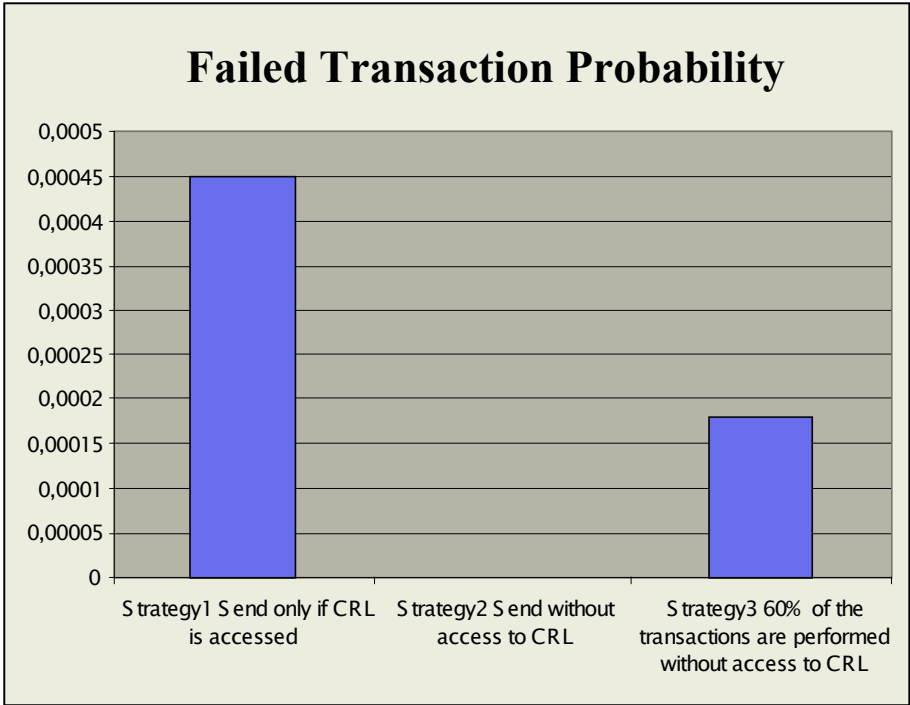


Fig. 2. Failed Transaction Probability for the three strategies.

It can be noticed that for strategy 1 the probability of security incidents to occur is extremely small although a large number of transactions may fail if the CA server is

unavailable. On the other hand if the transactions are performed without prior validation of the customer's certificate (strategy 2) there are no failed transactions but the possibility of a security incident increases significantly. In the case of strategy 3, according which the decision on whether the CRL will be accessed or not depends on the amount of the transaction and the threshold value F set by the financial institution, it is evident that it is possible to face both failed transactions and security incidents. However the probability of failed transactions is much less than the respective probability for strategy 1, while the probability of security incidents is less than the respective probability for strategy 2.

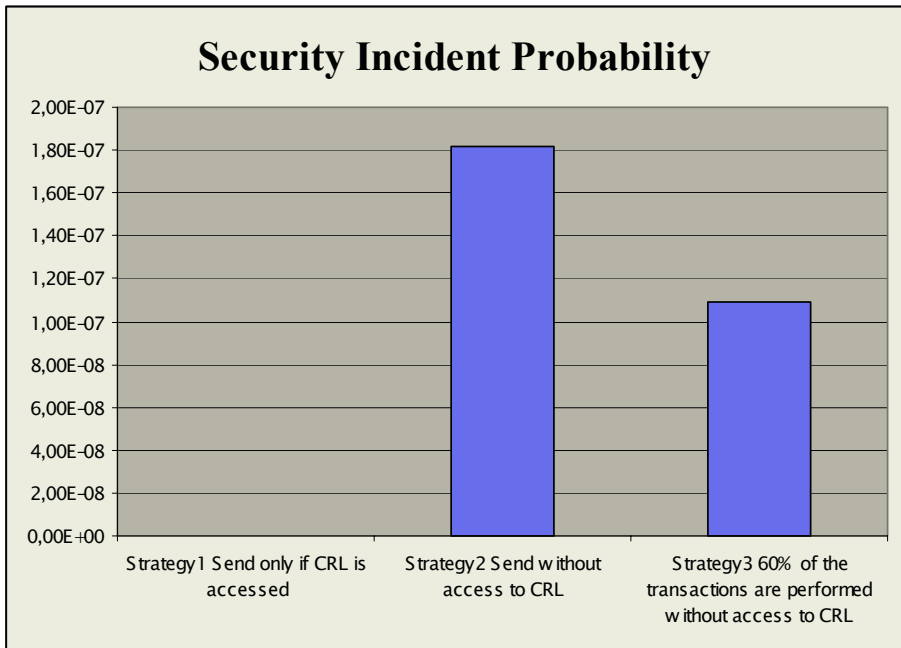


Fig. 3. Security Incident Probability as a Function of the Operational Strategy Adopted.

4 Expected Cost Due to Security Incidents and Failed Transactions (ECSIFT Indicator)

The steady state probabilities allow the evaluation of two undesirable events: the event of a *failed transaction* due to the unavailability of the CA server for verifying certificates' validity, which is a frequent event, and the event of a *security incident* which is a more exceptional event. On the other hand a failed transaction has a less significant cost compared to a security incident with a generalized impact.

The Markov modeling allows the evaluation of the probability of these events, although a more complete model taking into account additional parameters such as the derivation of the incident cost would be more appropriate for the evaluation of the operational safety of a PKI-based application.

For this purpose, the following probabilistic indicator is defined: The Expected Cost due to Security Incidents and Failed Transactions (ECSIFT) per unit of time.

If C_{SI} is the cost of a security incident and C_{FT} the cost of a failed transaction, the total probabilistic cost at time n is given as follows:

$$C_n = C_{SI} 1_{\{X_n=SI\}} + C_{LT} 1_{\{X_n=FT\}} \quad (3)$$

where X_n is the Markov chain modeling the system and $1_{\{\cdot\}}$ the indicator random variable.

In steady state, the probability of this event is given by $\lim_{n \rightarrow \infty} E[C_n]$, hence the expected cost is given as follows:

$$ECSIFT = C_{SI} \pi_{SI} + C_{FT} \pi_{FT} \quad (4)$$

Note that this probabilistic indicator can be derived from the general formulation of the performability indicator in [11].

The indicator ECSIFT (Expected Cost due to Security Incidents and Failed Transactions) for all three strategies is depicted in Fig. 4. For each strategy the ECSIFT has been calculated for two values of γ_1 (probability of CA server's unavailability), namely: $\gamma_1=0.001$ and $\gamma_1=0.002$.

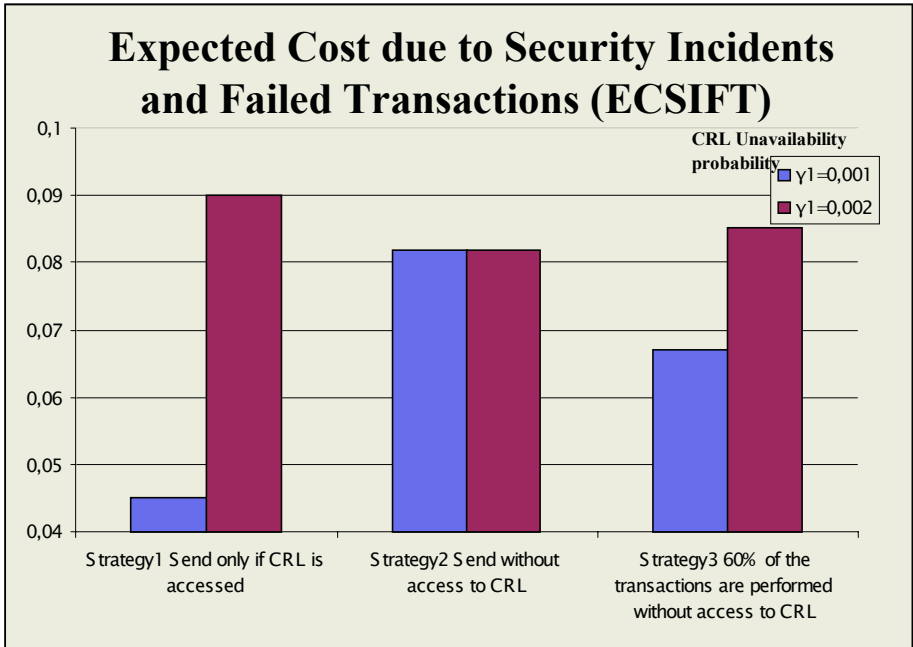


Fig. 4. ECSIFT Indicator for a CA Server's Unavailability Probability $\gamma_1=0.001$ and $\gamma_1=0.002$.

5 Discussion

Trying to comment the expected cost due to security incidents and failed transactions (depicted in Fig. 4), it can be noticed that if the probability of the CA server's unavailability is one time every 1000 transactions ($\gamma_1=0.001$) then strategy 1 seems to be the best approach while strategy 2 the most expensive one. Taking into account the fact that the cost of a security incident is assumed to be orders of magnitude bigger than that of a failed transaction, the ECSIFT figures for $\gamma=0.001$ are, somehow, the expected ones, since with strategy 1 there is an extremely low number of security incidents although a considerable number of transactions may fail due to unavailability of the CA server. With strategy 2 the situation is exactly the opposite since there are no failed transactions but there is a much bigger probability for security incidents.

However, as demonstrated in Fig. 4, a small differentiation in the probability of the CA server's unavailability may significantly alter the entire picture. More specifically if γ_1 becomes 0.002, implying that the number of failed transactions --as a result of the inability of the interested party to access the CRL-- will increase, then strategy 2 becomes the best choice, while strategy 1 the most expensive. This is because strategy 2 is not affected by the unavailability of the CA server as opposed to strategy 1 that will now face a significantly bigger number of failed transactions.

The main objective of this paper is to demonstrate the applicability and the practical advantages of the proposed probabilistic Markov model for PKI-based applications, which can be employed, even in a real-time fashion. Indeed, parameters such as the ratio of transactions exceeding a threshold cost, or the CRL unavailability can be directly obtained from a database tracing these specific time-varying parameters. The model is therefore supplied with updated input parameters allowing a dynamic decision-aided process concerning adoption of the less expensive operational strategy.

If more accurate results are required then more elaborated probabilistic models can be used. For instance, a strict assumption for using the Markov model is that the inter-arrival times of transaction requests are exponentially distributed, which however is not always the case for the sojourn time in the state where the CRL is unavailable. In such cases a semi-Markov model, cf.[7] is more suitable, allowing the consideration of any distribution for the sojourn time in the states.

Additionally, if a periodic behavior of some parameters is observed a non-homogeneous Markov model can be utilized. For instance, periodically, the number of requests for transactions that involve high amounts may reach a peak during some specific hours of a day; similarly the CRL unavailability probability may be variable during a day due to an overloaded server during these specific hours. In this case the ECSIFT indicator can be modeled with a cyclic non-homogeneous Markov chain, cf.[11], permitting a more accurate cost evaluation.

Some difficulties however subsist: the evaluation of some specific parameters such as the different costs:

- The cost of a failed transaction (C_{FT})
- The cost of a security incident (C_{SI})

Indeed, these costs are not easy to compute mainly because they depend on the application's environment but also due to lack of information. In this paper empirical data have been used in order to highlight the interest of the methodology, although *real* data would give more accurate results.

6 Conclusions and Future Work

The use of a PKI in large scale environments highlighted some inherent problems concerning the options to adopt for the optimal cost-centered operation of the system. A first approach is to always validate the digital certificates presented by the customers, in order to avoid, as much as possible, a *security incident*, even if this option may result in several *failed transactions* due to unavailability of the CA and thus of the CRL. A second option is to bypass the certificate validation step, aiming to minimize the number of *failed transactions* despite the higher probability of a *security incident* to occur. A third option is to access the CRL only for transaction amounts exceeding a certain threshold amount. The need to choose the optimal cost-centered solution led us to study the use of a Markov based probability model and specifically the introduction of a performability indicator, namely the ECSIFT, in order to evaluate the cost of each option in a decision support process. The results, despite the fact that there were obtained from empirical data, illustrated the need to use such models in environments with multiple operational and reliability parameters.

Our future work will be focused in identifying the cost-centered optimal transaction amount. For transactions exceeding this amount, the CRL check will be mandatory while in the remaining cases the CRL check will be optional.

References

1. Beaudry, M.: Performance related reliability for computer systems. IEEE Transactions on Computers, C-27, (1978) 540-547.
2. Ciardo, G., Marie, R., Sericola, B., Trivedi, K.S.: Performability analysis using semi-Markov reward processes. IEEE Transactions on Computers, C-39, (1990) 1251-1264.
3. Goyal, A., Tantawi A.N.: Evaluation of Performability for degradable computer systems. IEEE Transactions on Computers, C-36, (1987) 738-744.
4. Iyer, B.R., Donatiello L., Heidelberger P.: Analysis of Performability models of fault-tolerant systems. IEEE Transactions on Computers, C-35, (1986) 902-907.
5. Iliadis I., Spinellis D., Katsikas S., Preneel B., A Taxonomy of Certificate Status Information Mechanisms. Proceedings of Information Security Solutions Europe ISSE 2000, Barcelona, Spain, (2000).
6. Iliadis J., Gritzalis S., Spinellis D., Danny de Cock., Preneel B., Gritzalis D.: Towards a framework for evaluating certificate status information mechanisms. Computer Communications, 26(16), (2003).
7. Limnios, N., Oprisan, G.: Semi-Markov process and reliability, Birkhäuser (2001).
8. Meyer, J.F.: On evaluating the performability of degradable computing systems. IEEE Transactions on Computers, C-29, (1980) 720-731.
9. Pagès A., Gondran, M. : Fiabilité des Systèmes. Eyrolles, 1980.
10. Pattipati, K.R., Li, Y., Blom, H.A.P.: A unified framework for the performability evaluation of fault-tolerant computer systems. IEEE Transactions on Computers, C-42, (1993) 312-326.
11. Platis, A.: An extension of the performability measure and application in system reliability. Int. J. of Computational and Numerical Analysis and Applications 2 (1) (2002) 87-101.
12. Platis, A., Limnios, N., Le Du, M.: Performability of electrical power systems modeled by non-homogeneous Markov chains. IEEE Transactions on Reliability C-45 (1996) 605-610.
13. Platis, A., Tselios P., Vouros, G.: Attractability: an indicator for optimizing the design of Web sites. Datamining II, WIT Press (2000).

14. Rensburg, A., Solms, S., A reference framework for Certification Authorities / Trusted Third Parties, in L. Yngström and J. Carlsen (Eds.), Proceedings, IFIP 13th International Information Security Conference, Chapman & Hall, (1996).
15. Rivest R., Can We Eliminate Certificate Revocation Lists?. Proceedings of Financial Cryptography (1998).
16. Smith, R.M., Trivedi, K.S., Ramesh, A.V.: Performability analysis: measures, an algorithm and a case study. IEEE Transactions on Computers, C-37, (1988) 406-417.
17. Tsopelas, P., Platis, A.: Performability Indicators for the traffic analysis of wide area networks. Reliability Engineering and System Safety, 82 (2003), 1-9.
18. Van Moorsel, A.: Metrics for the Internet age: Quality of Experience and Quality of Business. HP Technical Report HPL-2001-179 (2001).
19. Wolter, K., Van Moorsel, A.: The relationship between Quality of Service and Business Metrics: monitoring, notification and optimization, HP Technical Report HPL-2001-96, (2001).

A Practical Approach of X.509 Attribute Certificate Framework as Support to Obtain Privilege Delegation*

Jose A. Montenegro and Fernando Moya

Computer Science Department, E.T.S. Ingenieria Informatica
Universidad de Malaga, Spain
{monte, fmoya}@lcc.uma.es

Abstract. This work introduces a particular implementation of the X.509 Attribute Certificate framework (Xac), presented in the ITU-T Recommendation. The implementation is based on the use of the Openssl library, that we have chosen for its advantages in comparison with other libraries. The paper also describes how the implementation is middleware-oriented, focusing on the delegation model specified by ITU-T proposal, and taking into consideration the ETSI report about Xac.

1 Introduction

Standard access control methods typically require that the stakeholder has privileges to access the computer resource. That type of systems use the underlying operating system for access control purposes, and require that all users of a shared resource are locally registered in the system. However, the requirement for individual system accounts on the computer system having the resource does not scale well.

Actually, there exist several systems where the authorization is controlled by elements that are external to the functional elements of the computer system. In this situation, it is necessary to establish a trusted relation among the functional elements and the elements responsible for the authorization.

This situation allows that several system share the same authorization system; hence, the global system is more scalable than traditional access control. This type of configuration is called *distributed authorization*, where the authorization system publishes the authorization elements and the functional system obtains those elements to evaluate the users' actions. Initially, the attributes or privileges belong to a functional system. These attributes are delegated to an authorization system, that manages them. Then, every functional system that trusts on the authorization system is able to use the delegated attributes.

* This work has been partially supported by the Spanish Ministry of Science and Technology under the Project TIC2003-08184-C02-01

We can find several proposals that try to resolve the above situation. However, they do not resolve completely the delegation problem, but constitute a good starting point to design a delegation system. These proposals are AAAARCH [10], Akenti [12], Kerberos [18], Sesame [9], and SDSI/SPKI [4]. As we will show, our implementation of the authorization framework is based on the ITU-T X.509 Recommendation, focusing on delegation functionality.

The rest of the paper is structured as follows. Section 2 presents the X.509 Attribute Certificate as the solution that conveys the authorization information preserving the security requirements. Section 3 describes the procedures used to add the X.509 attribute certificate to OpenSSL library, and the advantages to use this library. In section 4 we present the components of the middleware that is necessary to perform the framework and the protocol used to communicate AAs with clients. Finally, section 5 concludes the paper.

2 Authorization Information Structures

Authorization system needs to manage authorization information. This information require to use a format that provides authentication, confidentiality, integrity and non-repudiation. Actual solutions have adopted diverse formats to carry the authorization information like tokens, web cookies or identity certificates. These formats have several drawbacks, since they are not standard solutions or do not represent appropriate authorization information:

- *Proprietary solutions.*

Commercial solutions present their own format structure, like tokens or cookies, to store authorization information. Normally, proprietary formats present numerous bugs that produce security flaws in the whole system.

- *Attributes carried inside identity certificates.*

This solution does not make entity attributes independent from identity, what can cause problems. Firstly, this is not convenient in the frequent situations where the authority issuing the identity certificate is not the authority for the assignment of privileges. Secondly, even in the situations where the authority is the same one, we must consider that life of identity certificates is relatively long when compared to the frequency of change of user privileges. Therefore, every time privileges change it is necessary to revoke the identity certificate, and it is already known that certificate revocation is a costly process.

Moreover, many applications deal with authorization issues like delegation (conveyance of privilege from one entity that holds a privilege to another entity) or substitution (one user is temporarily substituted by another user, and this one holds the privileges of the first one for a certain period of time). Identity certificates support neither delegation nor substitution.

On the other hand, ITU-T firstly used the concept of *X.509 Attribute Certificate (Xac)* in 1997 [6]. Later, in 2000, a second version of *Xac* was proposed

to improve the previous version [7]. Then, the IETF produced a RFC [5] which recommended how to use the fields of the structure of *Xac* in Internet Scenarios.

This certificate structure is used, in our solution, to carry out the information between the elements that verify privileges of the entities. Moreover, it is the basic element to develop the *Privilege Management Infrastructure* (PMI).

3 Using the Openssl Library to Implement X.509 Attribute Certificate

Openssl is a cryptographic library based on SSLeay project. The SSLeay project was initialized by Eric A. Young and Tim J. Houston [19], and the initial proposal pretended to offer the SSL protocol for whatever security application. Openssl was created in 1995, and it is the cryptographic library most widely used. The main reasons are:

- It is distributed using an open source license. This allows users to make changes and optimizations in cryptographic algorithms.
- It is possible to integrate it in any commercial application without paying royalties, allowing to offer cryptographic-SSL support to any application comparable to commercial libraries.
- The library can be run in numerous platforms. This characteristic permits to export the applications that use the library to any operating system.

SSL protocol implementation requires diverse cryptographic elements. For instance, symmetric key algorithms (AES, DES, IDEA, RC5), asymmetric key algorithms (RSA, Diffie-Hellman, DSA), hash functions (MD5, SHA, SHA1), etc. The library also implements the elements necessary to management X.509 identity certificates, using ASN.1 [8].

The importance of this library is also due to its inclusion and use in several outstanding projects. The library is the core of Apache [15], the most widely used secure web server. Additionally, it is used by the web browser Opera [17] and in projects like OpenSSH [16], OpenCA [3], and XMLSec [11], etc.

Those reasons have influenced us to choose the OpenSSL library as the platform to develop our system. However, the actual version of OpenSSL does not provide support to *Xac*. Therefore, the very first task has been to include it in the library.

3.1 Including X.509 Attribute Certificates in OpenSSL

OpenSSL has two main parts: a cryptographic library (*crypto*) and a SSL library (*ssl*). These parts are divided into the source code and the dynamic output libraries, which are obtained after library compilation and linking.

Including *Xac* in OpenSSL only requires the modification of the cryptographic library because *crypto* defines the cryptographic algorithms and standards. Each cryptographic item is located in a directory that contains its necessary resources.

Thus, the whole process of including *Xac* into OpenSSL has been divided into the following three phases:

First Phase: Definition of the Certificate Structure This phase implements the *Xac* structure, translating the ASN.1 definition to C source code. The complete ASN.1 definition can be found in [7] and [5].

OpenSSL uses a macro mechanism to translate the ASN.1 scheme to code. For instance, the following structure is an example of how to define the issuer identification:

```
IssuerSerial ::= SEQUENCE {
    issuer      GeneralNames,
    serial      CertificateSerialNumber,
    issuerUID   UniqueIdentifier OPTIONAL
}
```

The result to apply the macro system to the above structure is the following C code:

```
ASN1_SEQUENCE(X509AT_ISSUER_SERIAL) = {
    ASN1_SIMPLE(X509AT_ISSUER_SERIAL, issuer, GENERAL_NAMES),
    ASN1_SIMPLE(X509AT_ISSUER_SERIAL, serial, ASN1_INTEGER),
    ASN1_OPT(X509AT_ISSUER_SERIAL, issuerUID, ASN1_BIT_STRING)
} ASN1_SEQUENCE_END(X509AT_ISSUER_SERIAL)

IMPLEMENT_ASN1_FUNCTIONS(X509AT_ISSUER_SERIAL)
```

Second Phase: Definition of the Associate Function In the previous phase, we have obtained a code structure. However, it is necessary to define a set of functions to access the *Xac* information from the user application.

The macro mechanism offers another advantage: it internally defines the functions used to encode the certificate structure into to DER codification [8]. OpenSSL codifies all their structures to DER before using them. The similarity between *Xic* and *Xac* allows to reuse the *Xic* functions, like hash, sign and verification ones. Thus, it is only necessary to include the high level functions.

The following example shows the set of functions that are used to include information in the certificate fields. The *X509AT* prefix depicts the category of the functions, and it facilitates developers to decide the appropriate functions to use.

```
X509AT_set_serialNumber(X509AT *x, ASN1_INTEGER *serial)
X509AT_set_notBeforeTime(X509AT *x, ASN1_GENERALIZEDTIME *tm)
X509AT_set_notAfterTime(X509AT *x, ASN1_GENERALIZEDTIME *tm)
X509AT_set_holder(X509AT *x, X509AT_HOLDER *holder)
```

Third Phase: Addition of the Elements to Compilation After performing the second phase, we have a new directory for compilation. This directory contains the *Xac* four source files: definition, structure certificate, function access certificate elements and high level functions.

The compilation needs the modification of some configuration files, as the files generated must be included. The *util/mkdef.pl* file contains the definition and *util/libeay.num* file contains the low and high level functions of *Xac*.

Finally, the compilation is performed using the methods defined in the library, and we get the library that provides *Xac* functions.

4 Middleware

This section describes a privilege management prototype that has been created by using, as a foundation platform, the previous *Xac* implementation in Openssl. The ITU-T proposal establishes four PMI models: General, Control, Delegation and Roles model. Our prototype pretends to include the four models, focusing on the Delegation model. Other implementations of the ITU-T proposal, like PERMIS project [1], focuses on Roles Model.

4.1 Basic Components of Middleware

As stated, our work is based on the ITU-T X.509 proposal, but is also influenced by the recent European Telecommunications Standards Institute (ETSI) report [14] about requirements for attribute certificates.

The base to implement the delegation is the existence of an attribute proprietary and how this entity passes to another entity the control over the attributes. The attribute proprietary is named *Source of Authorisation* (SOA) in the ITU-T Recommendation and *Attribute Granting Authority* (AGA) in the ETSI report.

The ETSI report explains, by using the figure 1, the elements that must be part of an Authorization System based on attribute certificates, and facilities the implementation of the system and the inclusion of the delegation concept.

Subscriber or Subject attribute registration allows the *SOA* or the *AGA* entities to introduce the attributes in the authorization system and specify, by using a policy, how to carry out the delegation process.

Our system is based on the elements described in following sections. The Authorization Authority module corresponds to the Attribute Registration Service and *Xac* Acquisition Service (figure 1).

The Authorisation Authority performs the following services: *Xac* Generation Service, Attribute Revocation Management Service and *Xac* Revocation Status Service.

Attribute Authority The *Attribute Authority* (AA) has been developed as a transactions server, using a client-server schema and thread-based. The server, principal thread, listens to ongoing client connections.

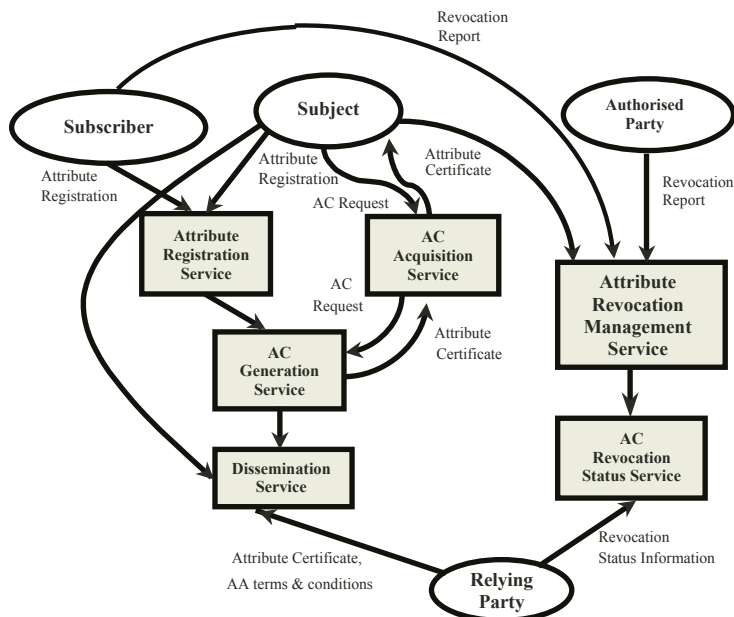


Fig. 1. ETSI: Components of an Authorization System based in Attribute Certificates

When a request is established, the client must be authenticated. The system supports several authentication mechanisms like traditional login-password, those based on symmetric or asymmetric cryptography, Ldap directories, Biometric, etc. The authentication mechanism can be extended with a plugin system implemented in the server.

Previous to a successful authentication, client and server must agree on the authentication methods. If the authentication process is successful, a session child thread is created and dedicated to attend the correspondent client. The figure 2 depicts the possible status of the AA.

The AA functions are related to the definition of the request that accept and process. The requests are composed of *Class*, *Function* and *Type*. The following statements are examples of request that an AA accepts (Types are not considered):

CLASS-CERTIFICATE: This class includes the request related to creation, signature and publication of certificates:

- *FUNCTION-GET-SIGNED* - Return the certificate signed by the AA.
- *FUNCTION-PUBLISH-LDAP* - Signs (if it is necessary) and publishes the certificate in LDAP

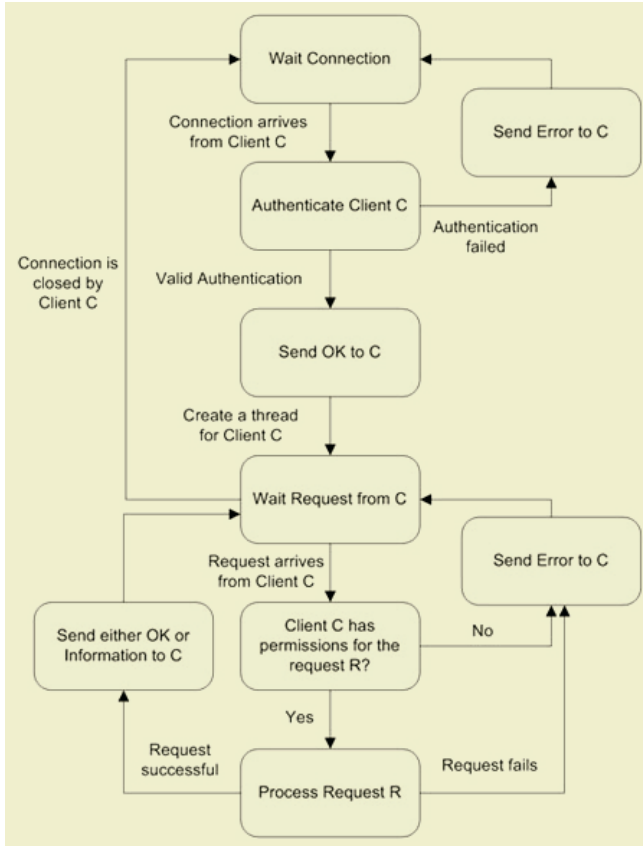


Fig. 2. Possible status of AA

CLASS-REVOCATION: This class includes the request related to certificates revocation

- *FUNCTION-REVOKE* - Revokes a certificate using its serial number as reference

CLASS-CONFIGURATION: This class includes the request related to the management of AA information

- *FUNCTION-ADD-ATTRIBUTE*, *FUNCTION-MODIFY-ATTRIBUTE* and *FUNCTION-REMOVE-ATTRIBUTE* - manages the lists of available attributes.
- *FUNCTION-ADD-EXTENSION* and *FUNCTION-REMOVE-EXTENSION* - manages the lists of available extensions.

- *FUNCTION-ADD-RESTRICTION*,
FUNCTION-MODIFY-RESTRICTION, and
FUNCTION-REMOVE-RESTRICTION - manages the lists of available restrictions.

CLASS-ERROR: This class is used to send error messages between entities

- *FUNCTION-DESCRIPTION* - Store the error message.

CLASS-REQUEST-INFORMATION: This class includes the request related to send information of the AA to the client

- *FUNCTION-FIND-ROID* - Sends to client a restriction reference to its *Object Identification*(OID)
- *FUNCTION-GET-FILE* - Sends to client different type of files. The file type depends on the function type used. The valid file types are available attributes, extensions, restriction and definitions.

In addition, the AA is in charge of the access and modifications of the list of attributes, extensions, restrictions and definitions, that are performed in a synchronized way. The transactions are registered in a log system to determinate the source of the operations.

Operator Attribute Authority The previous subsection defined the AA and the information exchanged with clients. This subsection presents a multiplatform communication module, which facilitates to developers the implementation to communicate the client and the AA.

The module can be used either as a part of noninteractive programs, or as part of interactive program. The visual interface facilitates the administration task to an Operator Attribute Authority. The figure 3 shows the GUI used for the configuration actions.

AA Client The AA Client sends the request to the AA. These requests are converted into AA decisions if the AA Client has the privileges needed. The AA defines and stores the authorized users and the tasks allowed for each authorized user.

There are different types to define the permissions:

- **CLASS:** Allows all request that include a class as type request.
- **CLASS | FUNCTION:** Allows only a function of the class.
- **PROFILE:** Allows only permissions defined in a determinate profile.

PROFILE is defined as a combination of permissions and actions. It is possible that a user has any combination of permissions and profiles.

This configuration allows to establish different figures in the infrastructure, each of them with an specific profile. For example, the following roles are available: Certificates Issuer, Certificate Revoker, Admin, etc.

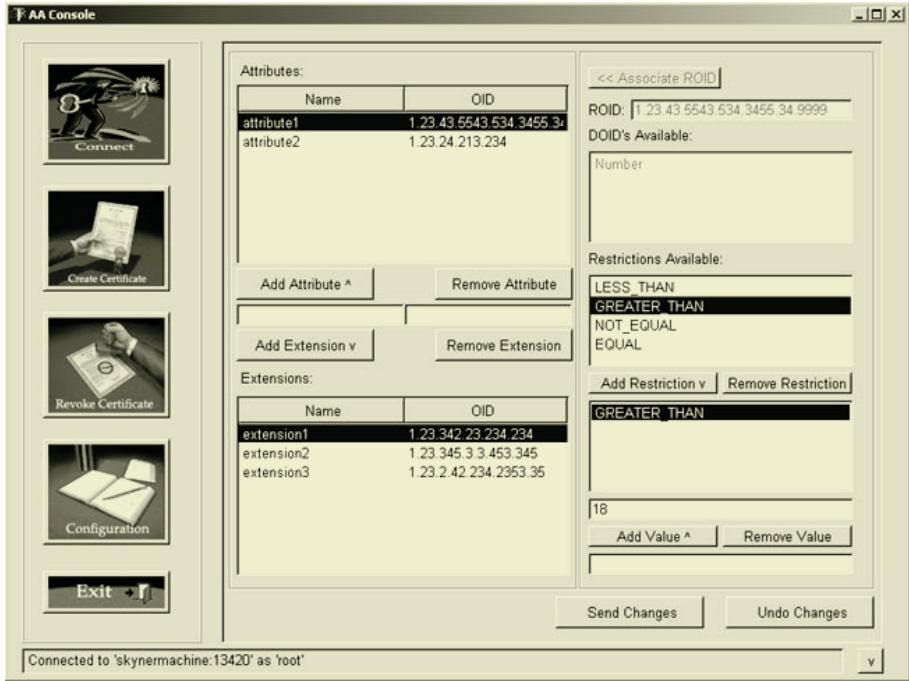


Fig. 3. Snapshot of AA Console

The following example depicts the possible profile configurations:

- *PROFILE-CERTIFICATE-GENERATOR*: CLASS-CERTIFICATE | FUNCTION-GET-SIGNED CLASS-CERTIFICATE | FUNCTION-PUBLISH-LDAP
- *PROFILE-CERTIFICATE-REVOKE*: CLASS-REVOCATION | FUNCTION-REVOKE
- *PROFILE-MANAGER*: PROFILE-CERTIFICATE-GENERATOR | PROFILE-CERTIFICATE-REVOKE

In addition, these profiles can be assigned to the AA Client:

- Manager: PROFILE-MANAGER
- Operator1: PROFILE-CERTIFICATE-GENERATOR
- Operator2: PROFILE-CERTIFICATE-GENERATOR
- Operator3: PROFILE-CERTIFICATE-REVOKE

The Client AA admin is established in dynamic form. This situation allows to create, modify or erase profiles without diminishing the AA functionality.

The preset Client schema facilitates the division of tasks since it is possible to assign enough privileges to users. For example, if the AA Client is a module of a web server that requires to generate a certain online attribute certificate, we must declare a certificate class:

- CLASS-CERTIFICATE
 - FUNCTION-GET-SIGNED
 - FUNCTION-PUBLISH-LDAP
 - *FUNCTION-GET-WEB-CERTIFICATE*: Issue an attribute certificate for certain user with five minutes validity.

In addition, the AA adds a web user:

- WebServerUser: CLASS-CERTIFICATE |
FUNCTION-GET-WEB-CERTIFICATE

Finally, the user only have access to the functions defined by the AA.

Storing Attribute Certificates in LDAP The *Xac* can not be treated in the same way as *Xic*. Normally an *Xac* must be linked to *Xic*, and an entity could have an unlimited *Xac*. This situation differs of the traditional solutions like *Xic*. Therefore, the unique element that distinguishes a *Xac* from another *Xac* is the serial number.

The serial number could be used to store the *Xac* in the directory; however, that is not the appropriate solution to retrieve the *Xac* because the serial number do not present information about the holder.

For that reason, the schema used to store and retrieve *Xac* in a Directory is that one defined in the draft [2], adding a mandatory field (MUST field) to definition (attributeCertificateAttribute):

```
(1.2.826.0.1.3344810.1.0.16
  NAME 'x509AC'
  SUP x509base
  STRUCTURAL
  MUST ( x509version
         x509serialNumber
         x509validityNotBefore
         x509validityNotAfter
         attributeCertificateAttribute)

  MAY ( x509acHolderPKCSerialNumber
        x509acHolderPKCissuerDN
        x509acHolderRfc822Name
        x509acHolderDnsName
        x509acHolderDN
        x509acHolderURI
```

```

x509acHolderIpAddress
x509acHolderRegisteredID
x509IssuerRfc822Name
x509IssuerDnsName

)

```

The reason to introduce the specified field is the difficulty to introduce every field as an individual entry. The modification allows that each LDAP entry is unique for each certificate.

The schema used to store and retrieve the *attribute certificate revocation list* (ACRL) is the specified in the standard [13]:

```

(2.5.6.19
  NAME 'cRLDistributionPoint'
  SUP top STRUCTURAL
  MUST (cn)
  MAY ( certificateRevocationList
        authorityRevocationList
        deltaRevocationList ))

(2.5.4.39
  NAME 'certificateRevocationList'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.9)

```

In this case, it has adopted the same solution as *Xic*, because both are revoked using the serial number.

4.2 Communication Protocol between User and AA

The communication protocol used has a simple design. The reason is that the security characteristics have been delegated to the SSL protocol. The SSL protocol provides confidentiality and data integrity, as well as client and server authentication, using *Xic*.

The protocol is based in a request/response method and has three phases:

1. *Connection Establishment*. During this phase the connection parameters are negotiated. If the client authenticates the AA, the connection is established.
2. *Identification*. The AA must authenticate the client. This authentication allows to establish the actions that a client can achieve. If the authentication succeeds, the protocol goes to phase three. Otherwise, the AA issues the corresponding error and ends the connection.
3. *Requests*. In this phase, a session is established between the client and the AA. Then, the client performs different requests to the AA (certificate creation, certificate publication, configuration task, etc). The AA response can be: confirmation, denegation or data exchange.

The connection is finished automatically and the session ends when the requests are completed or a timeout is reached.

The figure 4 depicts the elements necessary to realize the communication between the Client and the AA. The figure presents two possible views: *Virtual Connection*, between peers and *Real Connection*, between the AA and the Client.

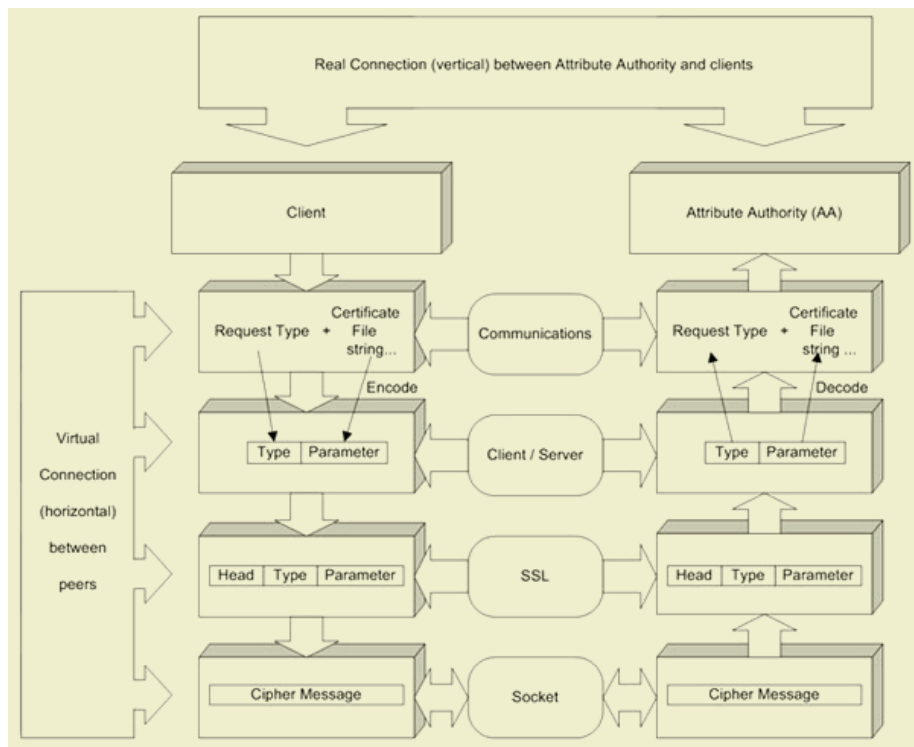


Fig. 4. Communication between AA and Client

5 Conclusion and Ongoing Work

The present work presents the basic elements needed to build an infrastructure that support delegation. The ITU-T proposal and the recent ETSI report have been used as a foundation to develop the prototype introduced.

At the moment of publishing the paper, only one library supporting *Xac* exists, this is IAIK. This library is implemented in JAVA language. The presented modifications to Openssl introduce another library that implements *Xac* in C

and C++ programming language. The inclusion of the functionality in Openssl library allows that the community uses the *Xac* provided as a free source code library.

Actually, the only *Xac* infrastructure known is that one from the PERMIS project. That project is based on the IAIK library for the implementation of the attributes certificates. The project has focused on role based access control infrastructures that use X.509 attribute certificates to store the users' roles.

Our initial studies about PERMIS project showed that the *Xac* generated in PERMIS project do not fulfill completely the standard. It introduces a slight modification in the *Xac* structure; more precisely in the *digestAlgorithm* entry of *ObjectDigestInfo* field. This variation affects the issuer and holder *Xac* fields. Contrarily, the solution we provide in the present work fulfills the standard defined by ITU-T and the IETF RFC.

References

1. D. Chadwick, "An X.509 Role-based Privilege Management Infrastructure", *Business Briefing: Global Infosecurity*, 2002
2. D. Chadwick, M. Sahalayev, "Internet X.509 Public Key Infrastructure LDAP Schema for X.509 Attribute Certificates", draft-ietf-pkix-ldap-ac-schema-00.txt.
3. C. Covell, M. Bell, "OpenCA Guides for 0.9.2+", <http://www.openca.org>
4. C. Ellison et al. "SPKI Certificate Theory", Request for Comments 2693, IETF SPKI Working Group, September 1999
5. S. Farrell, R. Housley, "An Internet Attribute Certificate Profile for Authorization", Request for Comments 3281, IETF PKIX Working Group, April 2002
6. ITU-T Recommendation X.509, "Information Technology - Open systems interconnection - The Directory: Authentication Framework", June 1997
7. ITU-T Recommendation X.509, "Information Technology - Open systems interconnection - The Directory: Public-key and attribute certificate frameworks", March 2000
8. B. Kaliski, "A Layman's Guide to a Subset of ASN.1, BER, and DER", November 1993
9. Tom Parker, Denis Pinkas, "Sesame v4 Overview", Issue 1. December 1995
10. J. Vollbrecht, P. Calhoun, S. Farrell, et al. "RFC 2904: AAA Authorization Framework", August 2000
11. A. Sanin, "XML Security Library Tutorial", <http://www.aleksey.com/xmlsec/>
12. Mary R. Thompson, Abdelilah Essiari, Srilekha Mudumbai, "Certificate-based Authorization Policy in a PKI Environment", TISSEC 2003
13. M. Wahl, "RFC 2256: A Summary of the X.500(96) User Schema for use with LDAPv3", December 1997
14. ETSI TS 102 158, "Electronic Signatures and Infrastructures (ESI); Policy requirements for Certification Service Providers issuing attribute certificates usable with Qualified certificates", V1.1.1, October 2003
15. <http://www.apache-ssl.org/>
16. <http://www.openssh.com/>
17. <http://www.opera.com/>
18. "Kerberos: The Network Authentication Protocol", <http://web.mit.edu/kerberos/>
19. <http://www.columbia.edu/~ariel/sslkey/>

TACAR: a Simple and Fast Way for Building Trust among PKIs

Diego R. Lopez¹, Chelo Malagon², and Licia Florio³

¹ RedIRIS. Edificio CICA, Av. Reina Mercedes, s/n E-41012 Seville (Spain)
`diego.lopez@rediris.es`

² RedIRIS. Serrano, 142 E-28006 Madrid (Spain)
`chelo.malagon@rediris.es`

³ TERENA. Singel 468 - D NL-1017 AW Amsterdam (The Netherlands)
`licia@terena.nl`

Abstract. The idea of setting up an on-line repository hosting the academic trust anchors arose within the TERENA Task Force for Authentication and Authorisation Coordination for Europe (TF-AACE) and gained immediately a great consensus within the academic community. Over the last months of the 2003 the TF-AACE group (promoted by TERENA) has formalized the policy, established a pilot site and exercised the procedures by incorporating several academic PKIs into the repository. The policy reflects the fact that the community of identity providers in the academic and research environment is a small one, and therefore personal trust relationships were already in place. The range of potential participants include National Research and Educational Networks (NRENs), National Academic PKIs in the TERENA member countries, and non-for-profit research projects directly involving the academic community. The first time an applying PKI asks to join the TACAR a face-to-face meeting between TERENA's representative and PKI's is required, in order to establish a sort of personal trustiness. Due to the fact that the certificates collected by the TACAR can be used for several purposes, the policy does not define minimum requirements for applying CAs and does not evaluate their CP/CPS against these requirements, but only establish which CAs can join the TACAR. Each organization using the TACAR is responsible for deciding which trust links it will establish. The TACAR is intended as a trusted source to obtain PKI root certificates enabling independent validation of trust links among different infrastructures.

Keywords: Certificate, policy, repository, trust link, trust anchor.

1 The Origins: the TF-AACE Initiative

In May 2002, the European NREN association, TERENA, launched TF-AACE (Task Force for Authentication and Authorization Coordination for Europe) [1] as follow-up of some meetings that were organized in the year before with the aim of exchanging information about the status of PKI technologies in the European academic networks. This was a coordination activity in the framework of

the TERENA middleware program, to encourage and support the cooperation between NRENs and other project teams in Europe in developing and deploying inter-operable authentication and authorization infrastructures (AAIs). Obviously, one of the key issues was the coordination of PKI activities.

PKIs have suffered of problems in reaching massive deployment among users, but play a key role in system interoperability. Although the use of public key infrastructure to identify final users has not happened yet (at least in great scale), it is clear that AAI components require sound foundations to establish trust links among them[2]. The simpler and more powerful choice is to apply a PKI to build these trust links. One of the first conclusions of the group was that any AAI spanning beyond campus limits (and even within them) requires a supporting PKI.

At the moment of starting the task force, PKIs have been already deployed (with varying degrees of completion) in several member networks, either in a national or regional basis, and even within specific communities of users. Actually, PKI technologies were intensively used by the Grid community, who had defined their particular manner to employ certificates for user authentication and, thus, had implemented specific solutions to their particular problems[3]. Anyway, the Grid community was (and currently is) the wider and more enthusiastic group of PKI users within academic networks.

Therefore, the group soon acknowledged the necessity of finding methods for integrating the different PKI initiatives, going further than traditional solutions. Despite the initial attempts, integrating the different existing PKIs in a common hierarchy found problems that made this solution impractical. These problems were mainly related to:

- Policies, as PKIs are intended for different purposes, depending on legal status of the operating institution, and are based on different principles.
- Technologies, as several (potential or actual) usages of PKIs impose limitations in the certificate verification procedures.
- The infrastructures themselves, as incorporating a PKI under a new structure requires a significative amount of re-signing tasks to let the user applications verify the new certificate chains.

Once the option of a common hierarchy was discarded, the group turned to the bridge approach[4], that offered very appealing characteristics and seemed to perfectly fit the highly heterogenous environment of European academic PKIs. However, bridges require a technologically complicated operating environment, and the potential benefits that could be acheived from their use were outweighed by:

- The difficult to find a feasible model for maintaining the intended European academic bridge CA, due to the costs derived from the model itself and from the high heterogeneity of the potential users.
- The small number of software elements (APIs, libraries, servers,...) able to deal with the complex trust paths derived from bridge CA usage, specially in the most generalized application domains.

- The lack of a real “killer application” compelling a fast adoption of this technology.

European academic PKIs seemed doomed to stay as a set of “trust islands”, just linked by scarce and ad-hoc bilateral agreements. This was the scenario when one of the TF-AACE members (Jan Meijer, from SURFnet) launched the idea of gathering the CA root certificates of all the participating NREN involved in the Task Force (and running a CA) in a single file, so it can be easily installed into user browsers. The idea was exhaustively discussed within the TF-AACE group and several implementations were proposed. The authors of this paper collected the different inputs and thanks to some others (most notably, Rodrigo Castro from RedIRIS and Reimer Karlsen from DFN) the idea of a repository for NREN CA root certificates gained momentum, as a trusted repository for certificates and policies that people administering PKIs (or any other network infrastructure relying on them) can use for assessing their trust models.

2 Means and Objectives of the TACAR

The TACAR (which stands for *TERENA Academic CA Repository*) is aimed to ease academic PKI interoperation by means of a trusted publication and distribution mechanism. Two main principles has guided its development:

1. “Keep it simple”: Do not require extra developments in the hosting or user organizations, enabling its sustainability.
2. “Let it happen”: Try to gain critical mass before making the system evolve to more complex structure and services.

Within TACAR, trust is established by means of the web of personal relationships among the participants in PKI-related initiatives coordinated by TERENA, or in which TERENA participates. The TACAR proposal intends to solve a PKI problem without PKI technology (somehow out-of-band). Intentionally, the initiative has started with the simplest possible approach, incorporating the minimal formalization required to guarantee that results are usable.

The process for incorporating a certificate (and its corresponding policy) within the repository is primarily based on face-to-face meetings. However, once the process has been started through the first direct meeting, PGP can be used as a trusted way to electronically exchange data for contributions or updates, thus avoiding paperwork and travels whenever possible.

Certificates in the TACAR are not supposed to be imported directly in final user browsers. The idea the repository is based upon is that the repository holds a centralized set of certificates and related CP/CPS, so an administrator at a given organization can analyze them, download the appropriate files and identify those certificates that the organization finds acceptable. From this moment, these certificates can be installed in applications, middleware components, and delivered to users in order to install them in their browsers.

The certificates collected are those directly managed by member NRENs, or belonging either to a National Academic PKI in the TERENA member countries (NPKIs), or to non-profit research projects directly involving the academic community. A policy document defining how to get involved, how the certificates are collected and made available on-line, and how the policy itself can be changed has been the first task of the TACAR team. This document, that defines the current operational status of the repository, is discussed in the next section.

TERENA, as the host organization of the repository, takes responsibility for undertaking the identification and authorization protocol and keeping the information in the repository up to date, but this is the limit of any responsibility that will be taken by TERENA. Compliance with the TACAR does not imply that the submitting body has passed any evaluation of its policy, but merely that the root CA was submitted to TERENA by bona-fide members of that organization who identified themselves to TERENA along with a legal recognized means of personal identification.

3 The TACAR Policy

The TACAR policy defines the steps for a certain PKI to be included in the repository, the procedures for updating the information pertaining this PKI, the measures to maintain the repository, and the process for changing the policy document itself. It also defines an accreditation process to allow for successive electronic updates of the data. The policy also includes as annexes template documents for the formal procedures defined in it. Its first version was released in January 2004.

3.1 Collection and Update of the CA Certificates

To bootstrap the entire process, a face-to-face meeting between a representative of the applying party and a TERENA officer has to be arranged. In this meeting, the applicant will provide the TERENA officer with two copies of a properly filled registration form (established by the policy document). If all the presented information is correct and mutual identity verification process is completed successfully (the policy mandates the use of photo-id documents), both the applying party representative and the TERENA officer will sign the documents. A CD-ROM with information pertaining the applying PKI has to be delivered together with the registration form. This data includes:

- The CA root certificates referenced in the registration form in TXT, DER and PEM format.
- A PDF version of the Certificate Policy (CP), if available
- A PDF version of the Certificate Practice Statement (CPS), if available.

At least one of the above policy documents must be provided to incorporate a PKI into the repository.

Collection of or updates to CA certificates can also be accomplished by electronic means, provided that the accreditation process (described in the following section) has been performed. This can be done by sending a mail message to the TERENA officer, including the properly filled registration form, digitally signed by one of the personal keys of the accredited persons.

Update of CA certificates is mandated whenever a change of the operating conditions occur (changes in the identity of the CA or the policy documents, revocation, abandon of service, etc.). Updates can be done either by means of accredited messages or by a face-to-face meeting. The possibility of critical updates being required makes the accreditation process, that allows for electronic updates, specially relevant.

3.2 The Accreditation Process

Any applying party willing to contribute to the TACAR may designate a number of individuals allowed to act as the institution contributors to the repository. Although this process is formally kept optional, it is highly recommend that applicants follow it. The applicant must provide to the TERENA officer, at any TERENA meeting, two copies of an accreditation letter (an example is included in the policy document). This letter designates a direct responsible person and at least two CA administrators, who will be in charge of registering, maintaining and updating the root certificate information on behalf of the applicant. The letter must contain information about the identity of the contributors and responsible people for the applicant, as well as the PGP personal keys to be used in the PKI data collection and update processes. The TERENA officer must positively identify (the policy mandates the use of photo-id documents) the person presenting the letter as affiliated to the applicant.

Once the first accreditation has taken place, any accredited person is able to add or remove people from the list of accredited individuals for the corresponding PKI. Updates can be done at a face-to-face meeting (presenting a new accreditation letter), by plain mail (as long as the update document is signed by any currently accredited person), or by electronic means, (including the new accreditation letter digitally signed by one of the personal keys listed in the previous one). The new accreditation information invalidates any previous accreditation.

3.3 Obligations of the Members and the Hosting Organization

Any organisation contributing to the TACAR is obliged to immediately inform TERENA of any update in its CA certificates, CP/CPS or any other information related to them in the repository (for example the link to the PKI site, the certificate download location, etc..).

TERENA, as responsible of the TACAR, is obliged to follow the identification/authorization protocol described in this document and to make certificates/policies and their updates publicly available. TERENA is also obliged to

check twice a year if the information in the repository is still correct by mailing the contributors. TERENA will decide on the base of the information got whether to add or remove certificates to or from TACAR.

TERENA provides a secured Web page allowing access to the file(s) conforming the repository, their MD5/SHA-1 fingerprint(s), and the fingerprint of any CA certificate contained in it. This Web page also provides access to each CP/CPS associated to any CA certificate in the repository, the MD5/SHA-1 fingerprint of the policies, and the links to the PKI sites. The TACAR certificate Web page is publicly available.

4 Applying the TACAR

The idea of setting up an on-line repository hosting the academic trust anchors has gained a significative consensus within the community. Over the last months of the 2003 the TF-AACE group has formalized the policy, established a pilot TACAR site and exercised the procedures by incorporating several academic and research PKIs into the repository. At the moment of this writing, five of these are available through the TACAR pilot site:

- CESNET (The NREN of the Czech Republic)
- GRNET (The NREN of Greece)
- RedIRIS (The NREN of Spain)
- SURFnet (The NREN of The Netherlands)
- DOEGrids (The Grid operated by the US DoE)

As stated in the TACAR policy, the repository in itself does not make any value propositions regarding the PKIs whose root certs it collects. Due to the fact that the certificates collected by TACAR can be used for several purposes, the policy does not define minimum requirements for applying CAs and do not evaluate their CP/CPS against these requirements, but only establish which CAs can join TACAR. As the repository begins to be used, some feedback, both about the repository itself and from the analysis of policies and the trust links decided as result of them, is expected.

There have been contacts with the European Policy Management Authority for Grid Authentication in e-Science (EUGridPMA [5]), an international body aimed to establish requirements and best practices for national e-science identity providers in order to enable common trust domains among them, in the spirit of PMAs[6]. In this context, the TACAR is envisaged as a trusted source to obtain PKI root certificates enabling independent validation of trust assertions spread via mechanisms like software distributions, very common in most Grid projects.

5 Conclusions and Future Work

The TACAR intends to be a pragmatic approach to the establishment of trust links among different PKIs that avoid the problems derived from the use of more

formal methods, such as a hierarchy or a bridge. The principles of simplicity and learning from experience has been key in the definition of the service, and the evolution of TACAR will be very much influenced by the needs of its user community. In any case, the interest that TACAR has raised within the academic and research community is quite promising.

As we have just said, the evolution of the service will be totally driven by user requirements as the TACAR is being applied. Nevertheless, some directions along which the service will be moving in the near future are:

- The definition of alternative procedures for downloading the certificates stored in the TACAR.
- The enhancement of the update procedures.
- The definition of methods for qualifying the trust links among different PKIs.
- The introduction of automated procedures (responders) for querying the repository.

References

1. TERENA, “TF-AACE. Authentication, Authorisation Coordination for Europe”, available at <http://www.terena.nl/tech/task-forces/tf-aace/>.
2. Gutmann P: “PKI: It’s not dead, just resting”. *Computer* vol.35 no. 8, August 2002.
3. Foster I, Kesselman C, Tuecke S: “The anatomy of the grid: Enabling scalable virtual organizations”. *International Journal of High Performance Computing Applications* vol. 15 no. 3, Fall 2001.
4. Alterman P: “The US Federal PKI and the Federal Bridge Certification Authority”. *Computer Networks* vol. 37 no. 6, December 2001.
5. EUGridPMA, “European Policy Management Authority for Grid Authentication”, available at <http://www.eugridpma.org/>.
6. International Grid Federation, “Grid Policy Management Authority”, available at <http://www.gridpma.org/>.

On the Synergy Between Certificate Verification Trees and PayTree-like Micropayments

Josep Domingo-Ferrer

Universitat Rovira i Virgili
Dept. of Computer Engineering and Mathematics
Av. Països Catalans 26
E-43007 Tarragona, Catalonia
jdomingo@etse.urv.es

Abstract. A substantial number of micropayment schemes in the literature are based on distributing the cost of a digital signature over several successive micropayments (*e.g.* Payword). Thus, a stable relationship between user and merchant is assumed: the micropayments validated by the same signature must take place between the same user and merchant. This stability is ill-suited for surfing on the web, a situation in which successive micropayments are made to different merchants. Thus coin-based micropayments, in which successive micropayments can be unrelated to one another, are far more interesting. One practical coin-based micropayment system is PayTree, which is amazingly similar to PKIs based on certificate verification trees (CVTs). We propose in this paper a synthesis of a CVT-based PKI with a PayTree-like micropayment system. The proposed system achieves a threefold reduction of the cost of public key certification and coin processing through: 1) sharing certificates by two applications (public key and multi-coin certification); 2) including certificates in the CVT on a batch basis (costs decrease as the batch size increases); 3) accelerating certificate (and thus public key and coin) verification through the verifier/merchant caching feature of the tree structure used. Furthermore, certificate and coin verification costs also become smaller as the number of users increases.

Keywords: PKI and eCommerce, Certificate Directories, Protocols, Coin-Based Payments, Micropayments, PayTree, Certificate Verification Trees.

1 Introduction

Micropayments are electronic payments of low value and they are called to playing a major role in the expansion of electronic commerce: example applications are phone call payments, access to non-free web pages, pay-per-view TV, etc. Standard electronic payment systems (like CyberCash [5], e-cash [8], iKP [2], SET [18]) for low-value payments suffer from too high transaction costs as compared to the amount of payments. The cost of transactions is kept high due to complex cryptographic protocols like digital signatures used for achieving a certain security level. However, micropayments do not need as much security as speed and simplicity (in terms of computing). For that reason and with few

exceptions [16,13,3], micropayment proposals try to replace the use of digital signatures with faster operations, like hash functions (whose computation is, according to figures by Rivest [17], about 10000 times faster than RSA signature computation).

Micropayment systems based on hash functions include NetCard [1], μ -iKP [10], PayWord [17], MicroMint [17], PayTree [11] and spending programs [6]. With the exception of MicroMint and PayTree, the above schemes use hash chains which represent chains of micropayments authenticated by a single signature on the first link of the hash chain. In this way, the cost of the digital signature is split over several successive micropayments. The price paid is that stable user-merchant relationships must be assumed: successive links of the hash chain can only be verified by the merchant who verified the signature on the first link. This stability assumption is shared by other micropayment schemes not using hash functions [16,13].

In MicroMint, the bank generates microcoins by computing collisions of a hash function. The appeal of this coin-based system is that successive micropayments can be quite unrelated. Thus one user may send one micropayment to merchant 1, the next micropayment to a different merchant 2, and so on. The problem with MicroMint is that its use of hash collisions requires very specific computational assumptions on the user, the merchant and the bank (see [19]).

PayTree uses a completely different approach to provide unrelated micropayments. It is a system based on Merkle's authentication tree scheme [14] and uses the following structure: a tree whose leaf nodes are labeled by secret random values, whose internal nodes are labeled by the hash value of the nodes' successors and whose root is signed. Because of the tree structure, PayTree is more flexible than PayWord-like systems in that it allows payments to different merchants to be made using different parts of the tree. Thus, multiple merchants can share the cost of a public-key signature.

While the need for micropayment systems assuming stable user-merchant relationships has recently been debated (why not use subscription or flat rate instead?, see [20]), the usefulness of unrelated micropayments for Internet surfing remains unquestioned.

1.1 Our Contribution

We propose in this paper to build a PayTree-like coin-based micropayment system upon an existing PKI based on certificate verification trees [9,7]. Due to the similarity between PayTree and CVTs, their combination in a single system results in a threefold reduction of the cost of public key certification and coin processing:

- The same certificates are used for two applications (public key and multi-coin certification);
- Certificates are included in the CVT in batches;
- Certificate (and thus public key and coin) verification benefits from verifier/merchant caching made possible by the tree structure used.

It is the low cost of coin processing what makes low coin denominations and thus micropayments possible with the proposed system.

In Section 2, we recall the advantages of CVTs with respect to other public-key certificate management systems. In Section 3, we describe our proposed solution. Section 4 analyzes the strong points of our proposal in terms of efficiency and security. Conclusions are summarized in Section 5.

2 Certificate Verification Trees

Like PayTree, CVTs are based on Merkle trees. Let a c-statement be a statement containing the name of an individual, her public key and an expiration date. In the CVT approach described in [9], a certification authority *CA* constructs a Merkle B-tree: each leaf is a c-statement plus the hash of that c-statement (obtained using a collision-free hash function). The hash values of siblings in the B-tree are hashed together to get a hash value for their parents node; this procedure iterates until the root node of the B-tree is obtained. Then *CA* signs the hash value of the root node, called *RV*, together with some additional information such as date and time. The *cert-path* for a c-statement is the path from the leaf containing the c-statement to the root, along with the hash values needed to verify that path (which include the hash values of siblings of nodes along that path). When a user requests a certificate of a public key, *CA* additionally supplies the cert-path of the c-statement, plus the signature on *RV*.

As new certificates are issued by *CA*, these are incorporated to the Merkle B-tree; tree update is performed on a regular basis, *e.g.* once a day. If the B-tree is a 2-3 tree, each certificate update requires only $O(\log n)$ work for the directory, where n is the total number of certificates. For each batch of certificates that is incorporated to the CVT, only one signature is computed on *RV*; the remaining computations are hashes, which using Rivest's figure are about 10000 times faster than signatures.

CVTs allow the *CA* key to be changed following compromise or just as a security measure, for example to increase the key length. *CA* just has to generate a new key pair, broadcast its new public key and sign the current *RV* with the new key. New *RV*'s arising from tree updates will be signed with the new private key as well. Note that in previous schemes where certificates are signed individually (like X.509, [4]), a *CA* key change requires to revoke all currently valid certificates, issue new certificates and forward those to their owners.

The ability to deal with historical queries is another strong point of CVTs. Certificate usage may be ephemeral (*e.g.* session authentication) or persistent (*e.g.* signature on a real-estate purchase). For persistent usage a requirement of non-repudiation appears: the user should be able to prove in some years from now that her public key was valid when using the certificate (*e.g.* when signing the purchase order). For historical queries to be possible with CVTs, *CA* should store the roots of the CVTs (one root per update period). For any contract needing persistent certificates, the cert-paths of the necessary c-statements should be stored with the contract, along with the signature on *RV*. In case

of *CA* key change, all stored root values should be signed with the new key. In previous schemes with individually signed certificates (*e.g.* X.509), allowing historical queries for persistent certificates requires *CA* to store all certificates ever issued, along with complete Certificate Revocation Lists (CRLs) for every update period; also a *CA* key change implies resigning all ever issued certificates and old CRLs!

Proving certificate non-existence is also an important feature of CVTs. CRLs and CRTs (Certificate Revocation Trees, [12,15]) do not provide evidence of the existence of non-revoked certificates. A CVT can comfortably store all certificates ever issued by a *CA*: using a CVT of height 30 (which yields cert-paths of length 30) is enough to store more than 10^9 certificates (which is more than the existing number of VISA credit cards). Thus, certificate forgery is difficult to hide.

Last but not least, CVTs give the possibility of verifier caching for efficiency. Verifiers checking many signatures every day (like vendors or routers using certificates for session authentication) may reduce communication with the CVT directory and computation by caching the top part of the CVT. This top part together with the *CA* signature on *RV* need only be retrieved and verified once per update period (*e.g.* once a day). Further, if a CVT has depth d and the top d_1 levels of it are cached, a signature verification will only require $d - d_1$ hashes.

The weak point of CVTs as used in [9] is that certificate issuance and revocation are synchronous: they are performed only at the start of each CVT update period. With daily updates at midnight, this means that a certificate request placed at 1:00 AM would have to wait 23 hours before being serviced. The same holds true for a revocation request. When comparing with the traditional solution of individually signed certificates, it becomes apparent that the improved manageability of the certificate directory is darkened by the delay in reacting to issuance and revocation requests. Assuming that the user is represented by a smart card, we presented in [7] a solution that maintains all advantages of CVTs while allowing for asynchronous certificate renewal and revocation. The idea of the scheme is that batches of public-key certificates can be requested ahead of time without the user having to store the corresponding private keys (which would increase the chances of key compromise). The user is represented by a smart card which performs all user functions. We will use that solution as the basis of the micropayment scheme in this paper.

3 Tree Micropayments in an Asynchronous CVT Public-Key Directory

The basic idea of the combination between micropayments and public-key certification is to certify coins together with public keys. From now on, the *CA* maintaining the CVT will also take the role of bank (we assume for simplicity that all users and merchants use the same bank). A certificate will contain a public key and several coins. Just as each certified public key corresponds to a *private* key needed to sign, each coin corresponds to a *secret* key needed to spend

the coin. The secret key of a coin is first revealed when the coin is spent during payment.

It is assumed that the user is *always* represented by a smart card SC or another tamper-resistant device. The scheme can be described by four protocols: certificate construction, certificate refreshment, payment and implicit revocation upon loss of user's smart card. In the description of those protocols, we distinguish communication between SC and CA from communication between SC and the CVT directory (for certificate download).

In the request protocol, a batch of m certificates is computed ahead of time by the user and stored by CA in the CVT. All certificates considered in the rest of this paper will be short-validity ones; more specifically, the i -th certificate in the batch contains a public key and n_i coins and is constructed to be valid during the i -th next CVT update period. Having established that successive certificates in a batch are assumed to be valid in consecutive CVT update periods, we will omit validity information in what follows in order to keep the notation simpler. Storing a certificate in the CVT by CA is an implicit CA signature on that certificate.

Protocol 1 (Certificate construction)

We assume that, in some set-up stage, the user's smart card SC has generated and installed a key k for a symmetric block cipher (e.g. DES, AES). All communications in this protocol are assumed to be authenticated (via shared-key or public-key encryption).

1. For $i = 1$ to m :
 - (a) SC generates a public-private key pair (pk_i, sk_i) .
 - (b) SC generates n_i coin secret keys, ck_1, \dots, ck_{n_i} and chooses n_i coin denominations d_1, \dots, d_{n_i} .
 - (c) SC computes $C_j = H(ck_j || d_j) || d_j$ for $j = 1, \dots, n_i$, where H is a one-way collision-free hash function and $||$ is the concatenation operator.
 - (d) SC encrypts $sk_i, ck_1, \dots, ck_{n_i}$ under k to obtain $E_k(sk_i, ck_1, \dots, ck_{n_i})$.
2. SC sends to CA the certificate $(pk_i, C_1, \dots, C_{n_i})$ and the secret information $E_k(sk_i, ck_1, \dots, ck_{n_i})$, for $i = 1$ to m .
3. SC deletes from its memory all items related to the batch of certificates (those items computed in Steps 1a through 1d).
4. In the next update of the general CVT, B adds

$$pk_i || C_1 || \dots || C_{n_i} || status || info$$

*to the CVT, for $i = 1$ to m . (Actually all coins sent by all users during the last update period are added to the CVT.) The field *status* contains as many bits as coins; initially, all bits are 0; in the next CVT update after C_j is spent, the j -th bit of *status* will be set to 1. The field *info* contains the rest of information usual in a certificate (owner, issuer, validity, etc.).*

In the above protocol, the choice of n_i depends on how many coins the user plans to spend during the validity period of the certificate. On the other hand,

the batch size m depends on the storage capacity of the smart card SC . The objective is for the user not to run out of valid certificates for the successive CVT validity periods; therefore, the larger m , the less frequently needs Protocol 1 to be run. Keeping this in mind, the user can go through the following protocol to obtain a current certificate at any time t :

Protocol 2 (Certificate refreshment at time t)

1. *If the user's smart card SC contains a certificate valid for time t then exit the protocol. (Note that, since short-validity certificates are used which are not supposed to survive more than one CVT update period, one never needs to refresh cert-paths for current certificates.)*
2. *Otherwise SC does:*
 - (a) *If the secret information of an expired certificate corresponding to a previous time t' is still in the card, delete from SC 's memory the private key $sk_{t'}$, stored and the coin secret keys related to the t' -th certificate, if any.*
 - (b) *Get from CA $E_k(sk_t, ck_1, \dots, ck_{n_t})$. Decrypt this value to get $sk_t, ck_1, \dots, ck_{n_t}$. (Before supplying the encrypted values to SC , CA in his role as a bank deducts from the account of SC 's owner the value $\sum_{j=1}^{n_t} d_j$ of coins contained in the t -th certificate. CA reimburses SC 's owner for the value of any unspent and deducted coins contained in expired certificates.)*
 - (c) *Get the certificate*

$$pk_t || C_1 || \dots || C_{n_t} || status || info$$

and its cert-path from the CVT directory.

Protocol 2 will be run each time a user wants to sign or pay. After completing Protocol 2, a user can sign using sk_t (which involves appending the certificate and the cert-path of pk_t to the signature). In order to pay, the user can spend the coins contained in the certificate for time t using the protocol below.

Protocol 3 (Payment)

Assume that the user has obtained a current certificate using Protocol 2 and wants to spend the j -th coin C_j contained in that certificate.

1. *The user sends to the merchant the whole certificate containing the coin*

$$pk_t || C_1 || \dots || C_j || \dots || C_{n_t} || status || info$$

as well as the cert-path of the certificate and the secret key ck_j corresponding to C_j .

2. *The merchant checks that C_j is included as an unspent coin in the certificate and that the certificate is included in the CVT directory (this latter check is equivalent to checking that C_j was certified by CA). This step can be simplified and communication with the CVT minimized as explained below.*

3. *The merchant checks the correctness of the secret key, by testing*

$$H(ck_j || d_j) || d_j \stackrel{?}{=} C_j$$

4. *If the checks in the previous step are OK, the merchant accepts a payment amounting d_j from the user.*
5. *Immediately before the next CVT update, the merchant sends all accepted coins to CA to redeem them. In his role as a bank, CA credits the merchant for the amount of those coins among those submitted by the merchant which had not yet been spent. In the next CVT update, all coins redeemed since the last CVT update will have their status changed from "unspent" to "spent" (this will result in a 1-bit change in the status field of the CVT leaves — certificates — containing those coins).*

The verifier/merchant can reduce her computation and communication needs at Step 2 of Protocol 3 (verification of a coin) by taking advantage of the verifier caching feature described in Section 2 for certificates in CVTs. If the merchant receives many payments every CVT update period (*e.g.* every day), she can cache the top part of the CVT. This top part together with CA's signature on the root value RV need only be retrieved and verified once per update period. Thus, if a CVT has depth d and the top d_1 levels of it are cached, a coin verification will only require $d - d_1$ hashes (verification of the $d - d_1$ lowest hashes of the cert-path of the certificate containing the coin) and no communication with the CVT.

If the SC is stolen or lost, the following revocation protocol is invoked by the user:

Protocol 4 (Revocation)

1. *The user informs CA on the loss of SC using some form of personal authentication (biometrical, handwritten signature via fax, etc.).*
2. *CA stops serving the encrypted information of certificates in future instances of Protocol 2.*
3. *CA marks the status of coins in future certificates submitted by SC as "spent". Those unspent coins marked as spent will be reimbursed by CA to the account of SC's owner.*

Protocol 4 does not revoke public keys nor coins till the next update period. However, if update periods are short enough, the thief will not be able to tamper with SC to have the card sign or run the payment protocol (Protocol 3) within the current CVT update period. For that assumption to be realistic, the card owner's identification must be moderately secure (*e.g.* a biometric procedure or a four-digit PIN with a limited number of typing attempts).

4 Performance Analysis

The advantages of the proposed CVT-based micropayment scheme are outlined in what follows. Note that the scheme maintains all of the advantages of asynchronous CVTs listed in Section 2 for generic certificates:

- The certification authority *CA* can easily change his key;
- Historical certificates (and thus historical coins) can be dealt with;
- Certificate (and thus coin) non-existence can be proven.

We next examine further strong points specifically related to coins.

4.1 Efficiency

The following are strong points regarding efficiency:

- *Shared cost of coin certification.* One public key and several coins are packed in a certificate and certificates are included in the CVT *in batches*. This reduces the per coin communication and authentication cost: *SC* sends multi-coin certificates in batches and the cert-paths are also returned by the CVT on a batch basis (Protocol 1). In addition, a single digital signature is computed by *CA* for all multi-coin certificates received within a CVT update period; thus, the cost of the digital signature is spread among all received coins and public keys to be certified.
- *Merchant caching for coin verification.* As explained in Section 3, a verifier/merchant verifying many public keys or receiving many coins (not necessarily from the same user) can save a lot of computation and verification by caching the top levels of the CVT during each update period. Thus the cost of running the payment protocol (Protocol 3) is also reduced by the use of CVTs.
- *Asynchronous coin certification.* Requesting certificates ahead of time allows *SC* to use as many public keys/coins as needed, regardless from the frequency of CVT updates (assuming that enough coins have been packed into the successive certificates).
- *Public status of coins.* The feature that all coins, whether unspent or spent, are visible in the CVT directory allows the status of a coin to be readily determined.

The fact that the cost of constructing and verifying certificates is shared by a batch of public keys and coins keeps the processing cost per coin and public key low. Beyond the batch effect, the cost is further reduced because the cost of the whole process is shared by two applications: public-key certification and micropayment. In this way, coins can have a low denomination because the cost of processing them is kept low.

4.2 Security

For a security discussion on public keys in asynchronous CVTs, see [7]. We concentrate in this section on the security for micropayments:

- *Coin counterfeiting.* Only coins included in the CVT are acceptable currency. For a user to be able to spend a coin this coin must have been included in the CVT packed in a certificate submitted by an *identified* smart card *SC*;

besides, the smart card owner must be backed by enough money in her account to recover the secret key of the coin (Step 2b of Protocol 2). Thus, there is no room for false coins.

- *Double spending.* The secret key corresponding to a coin must be supplied by a smart card *SC* for the coin to be acceptable. If the hash function is collision-free one-way, the secret key cannot be deduced from the certificate posted in the CVT. It remains to be shown that a coin cannot be spent twice (by the same user or by a merchant having received that coin legitimately). Clearly, a spent coin will be publicly marked as spent in the next CVT update. The problem is thus confined to double spending during the current CVT update period (*e.g.* during the current day). This type of double spending cannot be prevented but it can be tracked. Note that the CVT manager (*CA*) knows who ran Protocol 1 for the certificate containing a particular coin (the communication in Protocol 1 was an authenticated one). Thus:
 - When the bank is confronted to two redemption requests for the same coin, the two requestors and the user having requested the inclusion of that coin in the CVT are suspect (they all know the secret key of the coin).
 - Those three parties should explain what the coin flow was. If none of them confesses herself guilty, then the bank deducts the value of the double-spent coin from the account of each of them. In this way, the expected gain for double spenders is zero. Innocent punished parties will refrain in the future from accepting transactions from parties they had disputes with, which will add ostracism to zero-gain for real double-spenders.
- *Revocation of stolen coins.* If a smart card *SC* is lost or stolen and Protocol 4 is run by the owner of *SC*, there is a guarantee for the user that *SC* and the coins it stores will become useless from the next CVT update period onwards. As noted above, if update periods are short enough, the thief will not be able to tamper with *SC* to have the card run the payment protocol (Protocol 3). For that assumption to be realistic, the card owner authentication mechanism implemented in the card must be secure enough so that bypassing it takes longer than what remains of the current CVT update period.

5 Conclusion

This paper highlights some connections between public-key certificate directories and coin-based payment systems. Certificate verification trees are a very convenient data structure for managing large-scale public key directories and PayTree is a very convenient and flexible micropayment system. In this paper, we have shown how to exploit the CVT-PayTree similarity to integrate public-key certification and coin-based micropayments in a synergetic way that further reduces the processing cost per public key and per coin. In our proposal, certificates in the CVT pack one public key with a bunch of coins spendable during the

validity period of the certificate. To facilitate revocation, the validity period of certificates is short and matches the duration of a CVT update period.

The cost of public key certification and coin processing is reduced in three ways: 1) certificate sharing by two applications (public key and multi-coin certification); 2) certificate inclusion in the CVT on a batch basis; 3) lighter certificate (and thus public key and coin) verification thanks to the verifier/merchant caching feature of the tree structure used. The resulting scheme yields coin processing costs which become smaller as the batch size and the number of redeemed coins increase.

The smart card used for generating coins must be tamper-resistant, since it stores the secret information needed to sign and pay during the current CVT update period; at the same time, the owner identification mechanism built into the card should not be bypassable within a CVT update period.

Acknowledgment

This work was partly supported by the Spanish Ministry of Science and Technology and the European FEDER Fund under contract no. TIC2001-0633-C03-01 "STREAMOBILE".

References

1. R. Anderson, C. Manifavas and C. Sutherland, "NetCard - A practical electronic cash system", 1995. Available from author: Ross.Anderson@cl.cam.ac.uk
2. M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik and M. Waidner, "iKP - A family of secure electronic payments protocols", in *First USENIX Workshop on Electronic Commerce*, New York, July 1995.
3. M. Blaze, J. Ioannidis and A. D. Keromytis, "Offline micropayments without trusted hardware", in *Financial Cryptography'2001*, pp. 21-40, 2002. Springer-Verlag, LNCS 2339.
4. CCITT (Consultative Committee on International Telegraphy and Telephony), *Recommendation X.509: The Directory-Authentication Framework*, 1988.
5. CyberCash Inc., <http://www.cybercash.com>
6. J. Domingo-Ferrer and J. Herrera-Joancomartí, "Spending programs: A tool for flexible micropayments", in *Information Security-ISW'99*, pp. 1-13, 1999. Springer-Verlag, LNCS 1729.
7. J. Domingo-Ferrer, M. Alba and F. Sebé. Asynchronous large-scale certification based on certificate verification trees. In *Communications and Multimedia Security*, eds. R. Steinmetz, J. Dittmann and M. Steinebach, Norwell MA: Kluwer Academic Publishers, pp. 185-196, 2001.
8. e-cash, <http://www.digicash.com>
9. I. Gassko, P. S. Gemmell and P. MacKenzie, "Efficient and fresh certification", in *Public Key Cryptography'2000*, pp. 342-353, 2000. Springer-Verlag, LNCS 1751.
10. R. Hauser, M. Steiner and M. Waidner, "Micro-payments based on iKP", IBM Research Report 2791, presented also at SECURICOM'96. <http://www.zurich.ibm.com/Technology/Security/publications/1996/HSW96.ps.gz>

11. C. Jutla and M. Yung, "PayTree: "amortized-signature" for flexible micropayments", in *Proc. of the 2nd USENIX Workshop on Electronic Commerce*, pp. 213-221, 1996.
12. P. Kocher, "On certificate revocation and validation", in *Financial Cryptography'98*, pp. 172-177, 1998. Springer-Verlag, LNCS 1465.
13. M. S. Manasse, "The Millicent protocols for electronic commerce", in *Proc. of the 1st USENIX Workshop on Electronic Commerce*, July 1995.
14. R. Merkle, "A certified digital signature", in *Advances in Cryptology - Crypto'89*, pp. 218-238, 1990. Springer-Verlag, LNCS 435.
15. M. Naor and K. Nissim, "Certificate revocation and certificate update", in *Proceedings of 7th USENIX Security Symposium*, San Antonio TX, January 1998.
16. T. Poutanen, H. Hinton and M. Stumm, "NetCents: A lightweight protocol for secure micropayments", in *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, September 1998.
17. R.L. Rivest and A. Shamir, "PayWord and MicroMint: Two simple micropayment schemes", in *Security Protocols Workshop 1996*, pp. 69-87, 1997. Springer-Verlag, LNCS 1189.
18. Secure Electronic Transactions.
<http://www.mastercard.com/set/set.htm>
19. N. van Someren, "The practical problems of implementing MicroMint", in *Financial Cryptography'2001*, pp. 41-50, 2002. Springer-Verlag, LNCS 2339.
20. N. van Someren, A. Odlyzko, R. Rivest, T. Jones and D. Goldie-Scot, "Does anyone really need micropayments?", in *Financial Cryptography'2003*, pp. 69-76, 2003. Springer-Verlag, LNCS 2742.

A Socially Inspired Reputation Model^{*}

Nicola Mezzetti

Dept. of Computer Science, University of Bologna
7, Mura Anteo Zamboni
40127 Bologna, ITALY
`nicola.mezzetti@cs.unibo.it`
<http://www.cs.unibo.it/~mezzettin>

Abstract. In recent years we have observed a huge evolution of services deployed over the Internet, in particular by the World Wide Web; the development of Web Services has enabled Business-to-Business (B2B) relationships, i.e., the exchange of services over the Internet between different, possibly mutually distrustful, organisations.

Unfortunately, Web Services themselves do not provide all the features that are needed to implement and enforce B2B relationships; organisations have to be provided with security and trust guarantees so as to trust their services to be safely provided over the Internet.

In this paper we refine the trust foundations of [16] then we extend it with a model and metric of reputation inspired by the phenomenon of reputation in human societies so as to enable service consumers and producers to be mutually aware of their trustworthiness.

1 Introduction

In recent years there has been a rapid evolution of all Internet related technologies (e.g., Web Services, Business-to-Business technologies, middleware for Virtual Enterprises, etc.); now entities (e.g., service consumers and producers) are deployed in a global network context and interact with each other to implement complex behaviours of enterprise applications. For instance, terms such as *choreography* and *orchestration* have been introduced to deal with workflows of business processes [4,19] whose basic components are web services. Nowadays, not only web services have a role in wide area multi-party computations; mobile agents, components of geographically-distributed applications, and mobile devices are currently exposed to interactions between independent business parties. Unfortunately, it is not possible to know in advance whether a given entity can be trusted or not and the lack of a common trust semantics prevents applications from relying on a common trust infrastructure.

In such global computing environments, it is important to enable the scalable implementation of dynamic resource access control policies for supporting

^{*} This work has been partially supported by the European FP5 RTD project TAPAS (IST-2001-34069) and by the base project “WebMiNDS” under the FIRB program of the Italian Ministry of Education, University and Research.

the development of wide area distributed applications. Current research results address the problem by investigating models of trust in order to enable service producers and consumers to know how much reliance to justifiably place on each other; however, they prove to be inadequate to model a generic environment (see Sec. 7 and [1,2,3,5,6,7,9,10,12,15,18,20]).

In this paper we develop a reputation model to allow an entity to predict whether another entity will exhibit dependable behaviour or not, based on its past behaviour. First, we refine the foundations of trust presented in [16], then we extend it with a flexible model and metric of reputation that has been designed according to the social nature of trust. We expect this model to be generic enough to support reputation management within any possible global computing scenario; the examples provided in Sec. 4 justify our expectations.

This paper is structured as follows. Section 2 illustrates the foundations of trust. Section 3 describes the reputation model then, Sec. 4 presents a practical example of how it behaves. Section 5 introduces a novel abstraction, namely attributes, for improving the expressiveness of our model. Section 6 presents possible practical applications of a reputation model. Section 7 examines related research contributions. Finally, Sec. 8 concludes the work.

1.1 Trust, Trustworthiness, and Reputation

First, we define a *principal* to be an entity that can be granted access or affect access control decisions; the set of all principal is referred to as \mathcal{P} . A principal can consume or produce more than one interface; thus, it can be trusted within more than one context.

Second, we say Alice *trusts* Bob within a given context if Alice expects Bob to exhibit a dependable behaviour when acting within that context; thus, trust is a measure of how much reliance a *trustor* (e.g., Alice) can justifiably place on the dependability of a *trustee* (e.g., Bob) behaviour within a specific context. Although a *context* can be generically seen as any possible activity, we assume contexts to be associated with service interfaces because of the application of this research to global computing and B2B applications. To distinguish between principal's activities and, therefore, correctly compute trust degrees, each service interface i actually enables two contexts $p(i)$ and $c(i)$ that, respectively, indicate *provisioning* and *consumption* of that interface. Let i be any service interface then, for convenience in notation, we define $c(i) = p(i)^{-1} = (c(i)^{-1})^{-1}$ and vice versa.

Third, we say that an entity is trustworthy, within a given context, if it actually justifies reliance on the dependability of its behaviour within that context; thus, we define *trustworthiness* to be a private and secret property of any entity and, therefore, neither known to other entities nor provable.

Finally, we define *reputation* within a specific context to be the subjective perception of trustworthiness that is built from a given, possibly partial, amount of information that describe the behaviour that a specific entity exhibited in the past within the specified context.

2 Foundations of Trust

In this section we represent trust as a ternary relation $\mathcal{T}(\alpha, \beta, \phi)$, where α and β are two principals and ϕ is a context. The presence of the triple $(Alice, Bob, c)$ in \mathcal{T} indicates that Alice trusts Bob within context c .

A *trust system* is defined as the triple (P, Φ, \mathcal{T}) , where \mathcal{T} is the trust relation, P is the set of principals and Φ is the set of contexts on which \mathcal{T} is defined. A trust relation \mathcal{T} defined over $P \times P \times \Phi$ may satisfy the following properties:

reflexivity: A trust relation \mathcal{T} is reflexive in context ϕ if and only if, for every principal $\alpha \in P$, the triple (α, α, ϕ) has an associated trust value in \mathcal{T} .

symmetry: A trust relation \mathcal{T} is symmetric in context ϕ if and only if, for every pair of principals $\alpha, \beta \in P$, if $(\alpha, \beta, \phi) \in \mathcal{T}$ then $(\beta, \alpha, \phi^{-1}) \in \mathcal{T}$ (i.e., each has assessed the trustworthiness of the other).

Reflexivity is also called implicit trust; for example, within *authentication*, reflexivity expresses the trust that a principal places in the secrecy of its private key. More in general, trusting a principal within *authentication* means trusting the binding between his private key and his identity. We define *authentication* to be a special context that is equal to its inverse.

The symmetry property allows us to define the condition under which any two principals can communicate.

Proposition 1 (communicability condition). Let $\alpha, \beta \in P$ be any two principals, then a communication between them can take place if and only if $\mathcal{T}_{\{\alpha, \beta\}}$, the restriction of \mathcal{T} over the set of principals $\{\alpha, \beta\}$, is symmetrical in the context of authentication.

A third property, namely *transitivity*, is defined by introducing the *jurisdiction* subcontext; it is used to represent trustworthiness in recommending principals acting within a specific context. Given a context ϕ , the jurisdiction subcontext associated with ϕ is represented in mathematical notation by the symbol $j(\phi)$. A principal having jurisdiction over a context ϕ is trusted for providing reliable trust information about trustees within context ϕ (i.e., whether a given trustee can be trusted or not); such information is referred to as *recommendations*. For example, $(Alice, Bob, j(\phi)) \in \mathcal{T}$ means that Alice places trust in Bob for having jurisdiction over context ϕ and is willing to inherit Bob's trust relationships for context ϕ .

Transitivity is formally defined as follows:

transitivity: A trust relation \mathcal{T} is transitive in context ϕ if and only if for every three principals α, β, γ such that $(\alpha, \beta, j(\phi)) \in \mathcal{T}$ and $(\beta, \gamma, \phi) \in \mathcal{T}$ and the communicability condition holds for both $\{\alpha, \beta\}$ and $\{\beta, \gamma\}$, then $(\alpha, \gamma, \phi) \in \mathcal{T} \times \mathcal{T}$.

In other words, given a trust relation \mathcal{T} , transitive in context ϕ , and any three principals α, β, γ , if $\mathcal{T}(\alpha, \gamma, \phi)$ is not defined in \mathcal{T} , it can be indirectly computed if both $\mathcal{T}(\alpha, \beta, j(\phi))$ and $\mathcal{T}(\beta, \gamma, \phi)$ are defined; thus, the trust between principals α and γ is defined as $\mathcal{T}(\alpha, \gamma, \phi) = \mathcal{T}(\alpha, \beta, j(\phi)) \mathcal{T}(\beta, \gamma, \phi)$ in $\mathcal{T}_{\{\alpha, \beta, \gamma\}} \times \mathcal{T}_{\{\alpha, \beta, \gamma\}}$.

Table 1. Structure of Φ augmented with jurisdiction.

I	$::= i \mid h \mid k$
C	$::= p(I) \mid c(I) \mid \text{authentication}$
Φ	$::= C \mid j(C)$

In Tab. 1 the set Φ of the possible contexts is defined in terms of the definitions given above; here, I indicates the set of service interfaces and C the basic contexts.

2.1 Trust Degrees

So far, we have seen a trust relationship \mathcal{T} as a relation: given a couple of entities and a context, \mathcal{T} indicates the existence of a trust relation without giving any other information about the *strength* of this trust binding. In order to express this “strength”, the trust relationship is changed into a trust function that returns a real value belonging to the closed set $[0, 1]$ within which the value 1 indicates “full trust” and 0 indicates “absence of trust”; we name this strength measure *trust degree*. While remaining consistent with the trust definition given in Sec. 1.1, this representation greatly improves its expressiveness.

$$\mathcal{T} : \mathcal{P} \times \mathcal{P} \times \Phi \longrightarrow [0, 1]. \quad (1)$$

For example, $\mathcal{T}(\text{Alice}, \text{VISA}, p(\text{ATM})) = 0.99$ indicates that Alice expects from VISA’s automatic teller machine a dependable behaviour with a probability of 0.99, when used for withdrawing cash.

Trust degrees are not only used by a principal to decide whether or not to interact with a given principal; they might also be used to determine the security mechanisms and parameters that have to be required for the interaction to take place. For instance, in electronic commerce applications trust models are used to determine the frequency at which a principal has to authenticate performing micro-transactions payments [14].

Within transitive trust, the trustor should be prevented from trusting a given trustee more than both the trust he places on the recommender and the trust the recommender places on that trustee. Hence, if $\mathcal{T}(\text{Alice}, \text{Bob}, j(\phi))$ and $\mathcal{T}(\text{Bob}, \text{Cecilia}, \phi)$, then Alice would trust Cecilia in the same context with degree

$$\mathcal{T}(\text{Alice}, \text{Cecilia}, \phi) \leq \min\{\mathcal{T}(\text{Alice}, \text{Bob}, j(\phi)), \mathcal{T}(\text{Bob}, \text{Cecilia}, \phi)\} \quad (2)$$

where equality holds iff $\mathcal{T}(\text{Alice}, \text{Bob}, j(\phi)), \mathcal{T}(\text{Bob}, \text{Cecilia}, \phi) \in \{0, 1\}$. Inequality (2) is satisfied by defining *recommended trust* as the ordinary multiplication

between the trust degree that holds between the trustor and the recommender and the one that holds between the recommender and the trustee, within the appropriate contexts (see below).

$$\mathcal{T}(\text{Alice}, \text{Cecilia}, \phi) = \mathcal{T}(\text{Alice}, \text{Bob}, j(\phi)) \cdot \mathcal{T}(\text{Bob}, \text{Cecilia}, \phi) \quad (3)$$

3 Reputation Metric and Model

In this section we make use of the above described trust foundations in order to describe our social reputation model. To proceed further in defining reputation, we assume a trust system $(\mathcal{P}, \Phi, \mathcal{T})$ where $j(\phi)$ is defined for each context $\phi \in \Phi$.

So far, we considered trust a function that returns the trust degree between a trustor and a trustee within a given context. However, trust is not static in time; both time and events (e.g., an interaction that take place between the trustor and the trustee) can change a given trust degree. Thus, we define the trust function as below, where τ indicates the set of numerical time values.

$$\mathcal{T} : \mathcal{P} \times \mathcal{P} \times \Phi \times \tau \longrightarrow [0, 1] \quad (4)$$

First, since obsolete information is not considered to accurately describe more recent behaviours, we assume trust degrees decay as time passes; in order to consider trust information based on its freshness, we apply an exponential behaviour to the decay rate, $\eta(\phi) : \Phi \rightarrow [0, 1]$. In addition, the decay rate depends on the context in which the relation applies; the more critical (i.e., risky) the context is, the more rapidly the trust decreases with respect to time. Given a trust degree $\mathcal{T}(\alpha, \beta, \phi, t)$, computed at time t , the trust degree between α and β at time $t' \geq t$ without additional information is computed as in (5).

$$\mathcal{T}(\alpha, \beta, \phi, t') = \mathcal{T}(\alpha, \beta, \phi, t) \cdot \eta(\phi)^{t'-t} \quad (5)$$

Second, trust degrees (and reputations as well) dynamically adapt according to the interactions that take place in the system; i.e., a wrong or unexpected interaction result decreases the respective principals' trust degrees, and a good interaction result increases the respective principals' trust degrees. In (6) it is formalized how trust degrees are updated by information about new interactions.

We model the effect of an interaction on trust by evaluating its “trust value”, tv , ranging from 0 to 1. Let tv be the trust value associated with a new interaction that happens at time t' , the new trust degree is computed as a linear combination of both tv and the past value of $\mathcal{T}(\alpha, \beta, \phi, t)$; the ω parameter, namely the *trust stability* parameter, indicates how much the result of a new interaction affects a given trust degree.

$$\mathcal{T}(\alpha, \beta, \phi, t') = \omega \mathcal{T}(\alpha, \beta, \phi, t) \cdot \eta(\phi)^{t'-t} + (1 - \omega) tv \quad (6)$$

In our trust model, the function \mathcal{T} expresses the *direct trust degree* between a trustor and a trustee; unfortunately, it is not sufficient for us to represent

reputation. In fact, in order to define a social model of reputation, a set of recommendations from other principals has to be known. We define *indirect trust degree* to be the average trust-degree that a set Γ of known principals, namely *recommenders*, associate with the trustee (within a given context at a specific time). Indirect trust is formally defined as below; in that equation, it is worth noting the use of the notation $\mathcal{R}(\gamma, \beta, \phi, t)$ that stands for the recommendations obtained from other principals.

$$\mathcal{I}(\alpha, \beta, \phi, t) = \frac{\sum_{\gamma \in \Gamma} \mathcal{T}(\alpha, \gamma, j(\phi), t) \mathcal{R}(\gamma, \beta, \phi, t)}{\sum_{\gamma \in \Gamma} \mathcal{T}(\alpha, \gamma, j(\phi), t)} \quad (7)$$

The model described so far allows the subjective definition of *reputation* $\mathcal{R}(\alpha, \beta, \phi, t)$, as a linear combination in $[0, 1]$ of *direct trust* and *indirect trust*; it describes the trustworthiness that a trustor α associates with a trustee β within context ϕ at time t . Therefore, in (8) we formally define reputation; where the *trust balancing*, ψ , is a real parameter in $[0, 1]$ that indicates the subjective weight a specific principal assigns to direct trust with respect to indirect trust. It is individually decided by each principal.

$$\mathcal{R}(\alpha, \beta, \phi, t) = \psi \mathcal{T}(\alpha, \beta, \phi, t) + (1 - \psi) \mathcal{I}(\alpha, \beta, \phi, t) \quad (8)$$

Let any two principals carry out an interaction; after that interaction, each of them can associate a trust value with the other principal, according to his behaviour during that interaction. A new trust value does not only contribute to compute the direct trust between a trustor and a trustee; it is also used for computing the direct trust between the trustor and the recommendors. Basically, the smaller the difference between the recommendation and that interaction's associated trust value, the better reputation that will be associated with that recommender by the trustor. Thus, the trust value, tv_d , associated with that recommend will be 1 if the recommendation is equal to the interaction's trust value and will tend to 0 as the difference becomes larger; the new direct trust degree between the trustor and the recommender is to be computed according to (6) and tv_d .

3.1 Refining Trust Stability

Though it is clear that individuals form the direct trust about each other by combining information from their direct experience, how this information is combined is still unknown. Previously, we assumed direct trust to be the linear combination between the last computed direct trust and the fresh information. Intuitively, the more that the fresh information differs from the expected behaviour, the more that information is suspicious. Hence, the trust stability parameter is not to be a constant value; on the contrary, it should model possible reactions to anomalous behaviours. Trust stability does not have to uniquely depend on the distance between the last fresh value and the expected one; it also has to depend on whether this fresh value indicates a dependable behaviour or not.

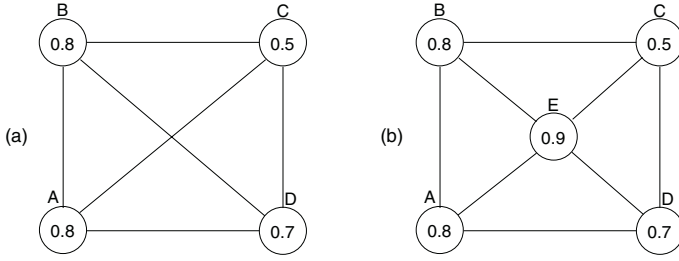


Fig. 1. Reputation in Peer-to-Peer network scenarios. In (a), a completely connected peer network is represented. Then, in order to increase the number of indirect trust relationships, in (b) a peer has been added.

In order for our model to be employed within some dependability context, we believe a cautious and responsible behaviour should be modeled by the trust stability. For instance, if the fresh trust value had meant an unusual low-dependable behaviour, then the trust stability value has to enforce a significant decrease the associated direct trust; in turn, if it had meant an unusual high-dependable behaviour, then the trust stability value has to enforce a negligible increase in the associated direct trust. Currently, we are simulating the behaviour of our trust model with different trust stability parameters so as to to optimize the trust-worthiness convergence time and to model cautious reactions to unexpected behaviours.

4 Setting Up a Reputation System from Scratch

To further illustrate the behaviour of our reputation model, in this section we describe how a reputation system can be implemented in the peer-to-peer network scenarios of Fig. 1. We represent a peer network as a graph where nodes indicate peers and links indicate the possibility for the peers to directly interact; trust information are exchanged along these links. In our graph representation each node is labeled with its trustworthiness; in order to make the example more comprehensible, we assume that each peer always (i.e., at every possible interaction within any possible context) exhibits a behaviour that is consistent with its own trustworthiness.

For the sake of conciseness, we focus this example on making the reader understand how reputations form within a distributed systems; for this reason, in this section we do not deal with the effects of time on reputations (i.e., we set the trust decay rate, $\eta(\phi)$, to be 1). The full document¹ provides examples of time-dependent behaviour of our reputation model.

Initially, each peer starts up without any information about the trustworthiness of the other peers (i.e., direct trust equal to zero). In addition, for each peer

¹ The full version of this document is available for download at the author's homepage.

Table 2. Reputations computed by B in the network scenarios of Fig. 1.

Scenario (a)				Scenario (b)				
run	A	C	D	run	A	C	D	E
1	0.422	0.265	0.368	1	0.381	0.252	0.130	0.481
2	0.636	0.398	0.556	2	0.568	0.339	0.188	0.717
3	0.731	0.457	0.640	3	0.648	0.388	0.216	0.823
4	0.772	0.483	0.676	4	0.682	0.409	0.229	0.869
5	0.789	0.493	0.690	5	0.696	0.418	0.235	0.887
6	0.795	0.497	0.696	6	0.702	0.422	0.237	0.895
7	0.798	0.499	0.698	7	0.704	0.423	0.238	0.898
8	0.799	0.499	0.699	8	0.705	0.424	0.239	0.899
9	0.799	0.499	0.699	9	0.706	0.424	0.239	0.899
10	0.799	0.499	0.699	10	0.706	0.424	0.239	0.899

we set the trust stability parameter and trust balancing parameters to be 0.4 and 0.6, respectively; these settings mean that each peer tends to forget past history in spite of recent experience and that it considers direct experience more relevant than the recommendations. Each peer collects recommendations by exchanging its own reputation table with its neighbors' ones; then, it will combine such collected information with trust derived by direct experience in order to compute the new reputations table.

We assume that the peers in the network collect trust information while interacting with each other. Hence, we define a protocol run as the set of actions by which a peer first (i) collects and updates information about direct trust, and then (ii) collects other peer's reputations to update his own reputation table; all the peers concurrently execute this protocol. The composition of the concurrent protocol runs that each peer locally exhibits enables the whole system to adaptively maintain peers' reputations. Thus, a protocol run can be defined as follows:

For each protocol run:

```
// direct trust update according to (6)
collect trust information and update direct trust;
// reputations update according to (7) and (8)
collect recommendations from neighbors;
compute new reputation table;
```

Table 2 shows the reputations computed by node B respectively in the scenarios (a) and (b) illustrated in Fig. 1. In the first scenario, where the peers are completely connected, it is possible to observe that B accurately approximates the reputation of the other peers with only 10 protocol runs (i.e., for each neighbor both 10 direct interactions and 10 reputation exchanges). Given a trustor and a trustee, the associated reputation also depends on the *social distance*, i.e.,

Table 3. Structure of Φ augmented with attributes.

I	$::=$	$i \mid h \mid k$
A	$::=$	$a \mid b \mid d$
G	$::=$	$p(I) \mid c(I)$
C	$::=$	$a(G) \mid G \mid \text{authentication}$
Φ	$::=$	$C \mid j(C)$

the average number of trust links that divide that trustor, or one of its trusted recommendors, from that given trustee; the higher the average distance between a trustee and a trustor is, the less precise information that trustor will be able to collect about about that trustee. Such information is then spread over the network, affecting the reputations computed by other peers. For instance, in the last scenario, it is possible to observe how the reputations computed by B are affected by the distances between C and A and between B and D.

5 Augmenting Contexts with Attributes

So far, we described how to evaluate a principal's reputation either in consuming or producing a given interface or in providing recommendations within a specified context. Hence, if we know that $\mathcal{R}(\text{Alice}, \text{Nicola}, p(\text{smtp}), t_0) = 0.67$, it is still unclear on which aspect the partial unreliability depends, e.g., whether Nicola's smtp server rarely allows illegitimate access to emails or just because of a low availability of the service. When a trustor finds a trustee's reputation to be unsatisfactory, in order for the trustor to interact with that trustee it is legitimate for him to be aware of the origin of that partial unreliability; attributes are introduced to fill this purpose.

We define an *attribute* to be a property that is relevant within a given context, such as availability, confidentiality or authentication. Attributes are introduced into our model by defining associated subcontexts, that are used to specialize the generic contexts.

In Tab. 3 the grammar describing the structure of Φ with attributes is formally described; here, I indicates the set of interface names, A the set of attribute names, G the set of simple contexts and C augments G with subcontexts.

Such a design approach enables us to simultaneously maintain both overall reputations and specialized ones in an efficient manner; in fact, when a principal is assigned a new trust value in a generic context $\phi \in \Phi$, if it corresponds to a specialized context, then the corresponding reputation can be updated and the associated generic context can be retrieved in order for it to be updated as well. Hence, the redefined definition of Φ allows us to manage specialized reputations

preserving the validity of the formulas presented in the previous section. The inverse is not defined for attributes.

6 Case Studies

So far, we have described our reputation model without discussing its possible employment. Clearly, it would not be practical for an application developer to encode reputation management within the application code; currently, an implementation of our reputation model is employed in the development of an application service for managing trust information on behalf of applications in a global computing scenario. Below, we introduce possible employments for implementations of our model of reputation. First, we introduce TAw [17], our trust-aware naming service that enables reputation management of entities (e.g., individual, resources, institutions) in distributed settings. Finally, we envision the employment of reputation models in the design of a hybrid trust management system that enables the efficient implementation of dynamical access control policies.

6.1 Trust-Aware Naming Service

Naming services identify a specific class of application services that enable the practical development of distributed applications as a collection of interacting resources (e.g., web-services, software components); basically, they implement location transparency of distributed resources by maintaining bindings between names and these resources. In a global computing setting, it is legitimate to assume those resources to be distributed in a multi-institutional scenario in which institutions are possibly mutually-distrustful. In such contexts, it is important for an entity (i.e., service consumer or producer) to be enabled with other entities reputations in order to refer to the most trustworthy provider of a given service or to trigger the appropriate security measures.

The Trust-Aware naming service (TAw) is a peer-to-peer middleware architecture designed to provide consumers with trust-awareness; TAw implements an abstraction layer between the actual naming service and applications. Such a middleware layer is implemented by a collection of TAw peers, one for each different entity in the system; each peer implements our trust semantics. A TAw peer is responsible for collecting trust informations, computing and disseminating reputations on behalf of its respective entity. Moreover, a peer is a client proxy between its associated entity and the naming service, providing that entity with trust-awareness with respect to other entities.

TAw represents our first attempt at integrating our reputation model into applications.

6.2 Reputation within RBAC

Role Based Access Control (RBAC) [13] is a credential-based trust management model that introduces the role abstraction to address the user-permission assignment, combining the advantages, in terms of efficiency, of both access lists

and capability access control models. In RBAC, permissions are assigned to role names and identities are granted to roles through role memberships; thus, the use of resources is restricted to individuals authorized to assume one of the legitimate roles. The RBAC model includes a Policy Description Language (PDL) that is employed for performing the user-roles assignment; a policy is a set of statements that define, for each role to be assigned which identity or roles ownership is to be proven. The use of roles to control access permissions can be an effective means for implementing institution-specific security policy in a flexible way, independent of the individuals that dynamically join or leave the institution.

Despite its flexibility, RBAC itself still does not efficiently enable the implementation of dynamical access control policies; in fact, in order to revoke a specific role membership or to adapt the access control policy, current RBAC implementations (e.g., [8]) rely on reliable group communication infrastructures for spreading access policy updates over the whole RBAC architecture.

We believe that our work can be employed for overcoming the aforementioned limitation of credential-based trust management systems; in fact, due to its adaptability and flexibility, our reputation model enables one to keep track of other principals' reputations and, thus, to evaluate whether a capability issued to him still has to be considered as valid. Still enforcing credential ownerships for a role to be activated, a reputation management infrastructure enables the definition of more complex PDLs; e.g., for any given role, it is possible to enforce a minimum reputation value to be maintained for a principal to be granted with and to maintain that role.

Reputation based techniques enable credential revocation to be efficiently implemented in other credential-based architectures.

7 Related Work

This work's principal contribution is a complete model of reputation that captures its social nature; thus, we provide a dynamical reputation model that is often partially undefined in related contributions. In addition, we introduce the novel attribute abstraction to improve the expressiveness of our model.

In [3], Aberer and Despotovic introduces a simple theory of trust to be applied in a document retrieval infrastructure; however, their model of trust neither supports trust transitivity nor considers time as a trust affecting factor.

In [5,6,7], Azzedin and Maherswaran develop a socially inspired reputation model that is similar to the one proposed in this paper. Here, it assumes that a trustworthy service provider in a given context has the same trustworthiness within recommending producers of the same service. To our point of view, this assumption cannot be made, in that it would be against the interests of that service producer; for this reason our model introduces the jurisdiction subcontext.

In [1,2], Abdul-Rahman and Hailes employs a reputation-based infrastructure for information retrieval. As in [3], trust does not depend on time.

The SECURE European Project is studying a reputation model specifically for ubiquitous computing [10]; they make use of knowledge degrees to disam-

bivariate between principals that present the same trust levels and to bootstrap the trust acquisition mechanism in an unknown environment. Although the trust model they develop is very precise about statical definitions, it is not clear how they develop the dynamical adaptation of trust and reputation degrees. By means of the decay functions that depend on time, our model encodes a degree of knowledge (i.e., a degree of the encoded information freshness) within the reputation model as well.

Both Dragovic *et al.* [12] and the OpenPrivacy project [18] implement reputation management systems; the former for trust-aware selection of application servers, and the latter for disseminating and computing trust information within a specific distributed object middleware. However, no specific information about the adopted reputation models were available to the authors.

In [9,20], Yahalom, Beth *et al.* describe a formal trust model, founded on bayesian probability theory and reliability analysis techniques, which is to be used for evaluating the trustworthiness of entities in open networks. In order for it to be applied, a trustor has to know all the paths to the trustee in the trust network, within the specific context in which trust is to be evaluated; hence, as the trust-network complexity and the trust contexts grow, this model does not meet scalability and efficiency requirements to enable its practical employment. Moreover, this model does not implement any technique for encoding the freshness of the available trust-information (e.g., trust decay as time passes). Our dynamical and adaptive reputation model does not require a trustor to handle any structured knowledge of the trust network; in addition, it is expected to meet the required flexibility and scalability requirements despite the network topology complexity.

7.1 The Web of Trust

In the context of Public Key Infrastructures (PKI), trust has been widely studied within the "Web of Trust" model [11,15]. Introduced by Zimmermann in Pretty Good Privacy (PGP)[21], the web of trust can be defined as PKI model where a global state about the principals and their direct trust relationships is maintained; i.e., each principal can make publicly known whose keys he trusts to be authentic. Moreover, as a member of the infrastructure, each principal can decide whom and to which degree to trust as an introducer of new keys. In order for communication to take place between any two principals, a trust link between them has to be enabled by the trust relationships encoded in the web of trust. In [15], Maurer formalizes the web of trust model and describes a technique for assessing probabilistic trust between two principals in a web of trust. However, such a technique require all the possible trust paths between the two principals to be known and a complex analysis technique to be applied on them; its complexity prevents such a technique to be applied in practice. In [11], Caronni introduces and evaluates heuristics techniques for reducing the complexity of the Maurer's technique; however, such techniques still require a global state about certificates and direct trust relationships to be maintained. Human Societies do not rely on having global knowledge of trust networks and,

still, the social reputation model succeeds in identifying untrustworthy individuals so as to support legitimate interactions among population members. Our social reputation model enables one to implement a reputation system without requiring principals to maintain a global state about the trust network, meeting scalability and efficiency even in arbitrarily complex scenarios. Moreover, our reputation model implements the dynamical behaviour that is not captured by the earlier model implemented in the web of trust.

8 Concluding Remarks

This paper's contributions are the refined trust semantics and their use in the definition of a socially inspired model of reputation. Moreover, the novel attribute abstraction provides a better degree of expressiveness with respect to other existing trust models. Despite the significant expressiveness improvement, the flexibility of our model is preserved; the presented example supports this result showing a possible use of our reputation model for setting up a trust infrastructure from scratch within peer network scenarios. We believe that our work is a step towards a better understanding of trust, trustworthiness and reputation from a computer science perspective.

Currently, we are developing our Trust-Aware naming service [17], namely TAw, that combines the presented reputation model with a gossiping based technique for trust dissemination. A discrete-event simulator of this architecture has been implemented and is being used for studying TAw's specific implementation of our reputation model as well as for studying its behaviour as the trust parameters change.

Acknowledgements The author thanks Licia Capra, David Hales, Vance Maverick, Lucian Wischik and the anonymous reviewers for the useful discussions and comments.

References

1. Abdul-Rahman, A., Hailes, S.: A Distributed Trust Model. In *Proceedings of the 1997 Workshop on New Security Paradigms*, pp. 48–60, Langdale, Cumbria, United Kingdom (1998)
2. Abdul-Rahman, A., Hailes, S.: Supporting Trust in Virtual Communities. In *Proceedings of 33rd Hawaii International Conference on System Sciences*, Volume 6, January 4–07 2000, Maui, Hawaii (2000)
3. Aberer, K., Despotovic, Z.: Managing Trust in a Peer-2-Peer Information System. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM'01)*, pp. 310–317, November 5–10 2001, New York, U.S.A., (2001)
4. Andrews, T., Curbera, F., Dholakia, H., Golland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: *Business Process Execution Language for Web Services (BPEL4WS) Specification 1.1*. IBM, BEA and Microsoft (2003)

5. Azzedin, F., Maheswaran, M.: Integrating Trust into Grid Resource Management Systems. In *Proceedings of 2002 International Conference on Parallel Processing (ICPP'02)*, pp. 47–54, August 18–21 2002, Vancouver, B.C., Canada (2002)
6. Azzedin, F., Maheswaran, M.: Evolving and Managing Trust in Grid Computing Systems. In *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'02)*, May 12–15 2002, Winnipeg, Manitoba, Canada (2002)
7. Azzedin, F., Maheswaran, M.: Towards Trust-Aware Resource Management in Grid Computing Systems. 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02), pp. 452–457, May 21–24 2002, Berlin, Germany (2002)
8. Bacon, J., Moody, K., Yao, W.: Access Control and Trust in the Use of Widely Distributed Services. *Software-Practice & Experience*, **33**(4) (April 2003), pp. 375–394 (2003)
9. Beth, T., Borcherding, M., Klein, B.: Valuation of trust in open networks. *Computer Security, Lecture Notes in Computer Science* vol. 875, pp. 3–18, published by Springer-Verlag (1994)
10. Carbone, M., Nielsen, M., Sassone, V.: A Formal Model for Trust in Dynamic Networks. In *Proceedings of Int. Conf. on Software Engineering and Formal Methods (SEFM 2003)*, pp. 54–61, A. Cerone and P. Lindsay (ed.), IEEE Computer Society Press (2003)
11. Caronni, G.: Walking the Web of Trust. In *Proceedings of IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'00)*, pp. 153–158, March 14–16 2000, Gaithersburg, Maryland, IEEE Computer Society Press (2000)
12. Dragovic, B., Hand, S., Harris, T., Kotsovinos, E., Twigg, A.: Managing Trust and Reputation in the XenoServer Open Platform. *Proceedings of the 1st International Conference on Trust Management*, May 2003, Heraklion, Crete, Greece (2003)
13. Ferraïolo, D., Shand, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: A Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security (TISSEC)* **4**(3) (August 2001), pp. 224–274 (2001)
14. Manchala, D.: E-Commerce Trust Metrics and Models. *IEEE Internet Computing* **4**(2), pp. 36–44 (2000)
15. Maurer, U.: Modelling a Public-Key Infrastructure. In *Proceedings of 1996 European Symposium on Research in Computer Security (ESORICS'96)*, E. Bertino (Ed.), *Lecture Notes in Computer Science* vol. 1146, pp. 325–350, published by Springer-Verlag (1996)
16. Mezzetti, N.: Towards a Model for Trust Relationships in Virtual Enterprises. In *Proceedings of 14th International Workshop on Database and Expert Systems Applications*, pp. 420–424, September 1–5 2003, Prague, Czech Republic, IEEE Computer Society Press (2003)
17. Mezzetti, N.: Enabling Trust-Awareness in Naming Services. Submitted for publication (2004)
18. The Openprivacy Project. <http://www.openprivacy.org/>
19. Peltz, C.: Web Service Orchestration and Choreography: a Look at WSCI and BP4WS. *IEEE Computer Journal* **36**(10), pp. 46–52 (2003)
20. Yahalom, R., Klein, B., Beth, T.: Trust Relationships in Secure Systems – A Distributed Authentication Perspective. In *Proceedings of 1993 IEEE Symposium on Security and Privacy*, pp. 150–164, May 24–26 1993, Oakland, CA (1993)
21. Zimmermann, P.R.: *The Official PGP Users Guide*. MIT Press, Boston, Massachusetts, U.S.A., (1995)

Using EMV Cards for Single Sign-On

Andreas Pashalidis and Chris J. Mitchell

Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom,
{A.Pashalidis, C.Mitchell}@rhul.ac.uk,
<http://www.isg.rhul.ac.uk>

Abstract. At present, network users have to manage a set of authentication credentials (usually a username/password pair) for every service with which they are registered. Single Sign-On (SSO) has been proposed as a solution to the usability, security and management implications of this situation. Under SSO, users authenticate themselves only once to an entity termed the ‘Authentication Service Provider’ (ASP) and are subsequently logged into disparate network Service Providers (SPs) without necessarily having to re-authenticate. The information about the user’s authentication status is handled between the ASP and the desired SP in a manner transparent to the user. In this paper we propose an SSO scheme where user authentication is based on payment cards conforming to the EMV industry standard. The card itself, in conjunction with the EMV architecture, takes the role of the ASP. The associated SSO protocol does not require online card issuer participation, preserves user mobility and does not put user’s financial data at risk.

Keywords: single sign-on, EMV, authentication

1 Introduction

Network users typically have to manage a set of authentication credentials (usually a username/password pair) for every Service Provider¹ (SP) with which they are registered. The number of such SPs with which a typical user interacts has grown beyond the point at which most users can memorise the required credentials. The most common solution is for users to use the same password with every SP — a tradeoff between security and usability in favour of the latter.

Single Sign-On (SSO) has been proposed as a solution to the usability, security and management implications of this situation. It is a technique whereby users authenticate themselves only once to an entity called an *Authentication Service Provider* (ASP) and are logged into the SPs they subsequently use, without necessarily having to re-authenticate. This seamless experience increases the usability of the network as a whole but introduces a number of security requirements. It is obvious that, under SSO, SPs require some kind of notification from

¹ In the context of this paper a service provider is any entity that provides some kind of service or content to a user. Examples of SPs include messenger services, FTP sites, web sites and streaming media providers.

the ASP about the user's authentication status. These notifications are termed *authentication assertions*. The SP, based on the authentication assertions provided by the ASP, determines whether or not to grant access to a protected resource to the specified user.

EMVCo², an organisation formed by Mastercard³ and Visa⁴, has developed a set of Integrated Circuit Card (ICC) specifications for Payment Systems [1, 2, 3, 4] that focus on the card/terminal interactions that take place at the Point of Sale between a cardholder and a merchant during a financial transaction.

In this paper we present a scheme in which EMV-compliant cards provide user (i.e. cardholder) authentication, and propose an associated protocol that facilitates SSO at disparate network SPs. In this scheme the cardholder's network access device itself, in conjunction with the card, acts as the ASP. The scheme can be regarded as an alternative to other smartcard-based authentication schemes, for example schemes that rely on Subscriber Identity Module cards (used in cellular telephony). The paper is organised as follows. The next section is a review of relevant EMV architectural components and security services. Section 3 describes the proposed authentication method and SSO protocol, while section 4 analyses the associated security threats. Section 5 discusses advantages and disadvantages, and sections 6 and 7 give an overview of related work and conclude the paper.

2 Review of EMV Security Services

This section introduces those components of the EMV specification that are relevant to this paper. For a full description see [1, 2, 3, 4, 5].

In the EMV payment system there are four major interacting entities, namely the cardholder, the merchant, an acquiring bank (the Acquirer) and the card issuing bank (the Issuer). The specifications focus on the interactions between card and merchant terminal. When the card is inserted into the terminal, the steps that occur include the following.

1. The terminal selects the appropriate EMV application by issuing a SELECT command [1, p.65] to the card.
2. The terminal initiates 'Application Processing' by issuing a GET PROCESSING OPTIONS [3, p.19] and a number of READ RECORD [3, p.23] commands. The purpose of this step is for the card and the terminal to exchange the necessary data for the rest of the transaction.
3. The terminal performs 'Processing Restrictions' [3, p.48]. This mandatory step does not involve communication with the card — its sole purpose is to provide assurance of application compatibility between terminal and card.
4. The terminal issues an INTERNAL AUTHENTICATE [3, p.21] command to the card. This optional step initiates 'Offline Data Authentication', which

² <http://www.emvco.org>

³ <http://www.mastercard.com>

⁴ <http://www.visa.com>

can be either Static Data Authentication [2, p.15] or Dynamic Data Authentication [2, p.24]. The purpose of this step is to verify the card's authenticity.

5. The terminal performs 'Cardholder Verification' [3, p.50]. During this optional step the cardholder's Personal Identification Number (PIN) is verified, either offline to the card (using the VERIFY command [3, p.25]), or online to the Issuer.

We next focus on the Dynamic Data Authentication (step 4) and the Cardholder Verification (step 5) that are defined in the EMV specifications. These steps are of particular interest since, in the authentication/SSO scheme proposed below, the card reader is under the control of the cardholder.

2.1 Dynamic Data Authentication (DDA)

DDA is supported by a Public Key Infrastructure (PKI), as specified in [2]. In particular, every DDA-capable card has its own asymmetric key pair which is generated and loaded on the card by the Issuer. While the private key cannot be extracted from the card, its public counterpart can be retrieved with a READ RECORD command. This public key is embedded in a public key certificate which is signed by the Issuer. The Issuer's public key certificate, signed by the Payment System's top-level Certification Authority (CA), is also stored in the card and can be retrieved by the terminal. As a result, the merchant terminal only needs to maintain an accurate copy of the CA's trusted root public key in order to verify the Issuer's, and hence the card's, public key certificates, and finally any data signed by the card itself. CA public key management principles and policies are defined in [2].

A simplified description of Dynamic Data Authentication (DDA) is given below:

1. The terminal retrieves the Issuer and card public key certificates from the card. The latter is verified using the former and the former is verified using the appropriate trusted root public key of the Payment System's top level CA.
2. The terminal issues an INTERNAL AUTHENTICATE command to the card. The command requires a number of parameters, including a nonce.
3. The card computes a digital signature over the terminal-provided data (including the nonce) and 'card Dynamic Data', which is data generated by and/or stored in the card [2, p.35]. The card outputs the signature and the card Dynamic Data.
4. The terminal reconstructs the signed data structure and verifies the signature using the card's public key retrieved and verified in step 1.

DDA provides data integrity and freshness. Assuming tamper resistance of the card and the soundness of the Issuer's security procedures, DDA also provides card authentication and card counterfeiting prevention. It should be noted that not all EMV-compatible cards are DDA-capable.

2.2 Cardholder Verification

The identity of the cardholder is verified based on a PIN. The PIN is entered into the terminal and may then either be verified online to the Issuer or offline to the card. In the latter case the terminal issues a VERIFY command to the card which takes the PIN as a parameter. It may or may not be encrypted using the card's public key. The card checks whether the supplied PIN matches the one stored inside the card and responds accordingly. If the number of unsuccessful offline PIN verification attempts exceeds a certain limit, the PIN on the card is blocked and can only be unblocked using a script sent to the card by the Issuer.

3 Using EMV Cards for SSO

This section describes the proposed EMV-based SSO scheme.

3.1 System Entities

The entities involved in the authentication/SSO scheme are the cardholder system, the card itself, and the SPs.

As briefly mentioned above, the system requires the cardholder system and the card to collectively act as the ASP. Instead of directly authenticating to every SP, the cardholder is authenticated by the card, and the card then vouches for the identity of the cardholder to every SP. The fact that the CS would typically consist of a 'standard' PC, PDA or mobile phone equipped only with a special SSO application and an EMV card means that this offers an inherently mobile SSO solution; the SSO application could be downloaded from a trusted source when required, and the EMV card is inherently mobile.

The Cardholder System The Cardholder System (CS) consists of the user's (i.e. the cardholder's) network access device and a card reader. A typical configuration would be a PC with an integrated card reader. Whether or not the card reader is equipped with its own (trusted) keypad is optional (see section 4.4). Alternatively, the CS could be a wireless network access device (such as a Personal Digital Assistant (PDA) or a 3GPP⁵ mobile phone) capable of communicating with EMV cards. The CS also needs some special software that implements the SSO protocol described in section 3.3 below. This 'SSO agent' might be realised as a process that continually runs on the CS (also known as 'service' or 'daemon'), or as part of the software that is used to access the SP (e.g. the web browser, instant messenger, e-mail client, etc.). In this latter context the SSO agent could be uploaded to the CS as an applet running within the SP access software, e.g. as a Java applet running within the web browser, or a Java MIDlet that is delivered over-the-air to a mobile phone. The SSO agent is likely to be provided by an EMV card issuer or a trusted third party.

⁵ <http://www.3gpp.org>

The Card The proposed EMV-based SSO scheme imposes certain requirements on the cards, as follows. Cards must be DDA-capable. Unfortunately, from the SSO perspective, the public key certificate used during DDA binds the card's public key to the cardholder's Primary Account Number [2, p.33]. It would constitute a potentially serious security and privacy threat if the Primary Account Number was to be used in an open environment such as the Internet. Therefore, the cards used in the scheme described here need to possess a separate, dedicated EMV application, which we call the *card authentication application* (AA). In the AA, the Primary Account Number (and any other personally identifying information) must be replaced with data that is not linked to the cardholder's identity (and cannot be used for financial transactions). This implies that the Issuer has to provide the AA with an additional certificate for the card's public key. It is required that this certificate does not contain any personally identifying information for the cardholder; thus we call it an 'anonymous certificate' in the sequel. As its serial number can be used for user identification at SPs, it should not contain any other information about the user (e.g. a name). Furthermore, the AA should be able to maintain state within the current session. In particular, it should be able to maintain a data element that indicates whether or not offline PIN verification (via the VERIFY command) has been performed during the current session, and, if so, the data element should also indicate whether or not PIN verification was successful. This card-provided *PIN Verification Data Element* (PVDE) shall be included in the data that is signed by the card during DDA, as part of the card Dynamic Data.

It should be noted here that a card session begins with Application Selection (step 1 in section 2) and ends when the card reading device deactivates the card [1, p.17]. This latter event includes premature removal of the card from the reader.

Service Providers In the proposed SSO scheme, SPs are required to accurately obtain and store the root keys of the CAs of the EMV Payment Systems that are to be supported (and trusted). This requirement is exactly the same as that applying to merchant terminals for 'standard' use of EMV cards. The management, distribution and revocation of these root keys is outside of the scope of this paper, but the principles are similar to those specified in [2] for merchant terminals. It is assumed that SPs require a user to be authenticated before granting access to protected resources. Instead of executing an authentication protocol directly with the user, SPs acquire the necessary authentication assertions from the CS, according to the protocol described in section 3.3 below. Moreover, as users also need to authenticate SPs, it is necessary that every SP possesses a *unique, human-readable identifier* (SPID).

3.2 Trust Relationships

The SSO scheme depends on the EMV cards offering a level of tamper-resistance, since these cards act as a trusted computing module within the CS. In addition,

cardholders need to trust that SPs will not collude in order to compromise their privacy (see section 4.1). Cardholders and SPs also need to trust that

- the Payment System’s top-level CA(s) will not impersonate cardholders,
- card Issuers will not impersonate cardholders.

From the cardholder perspective, authentication/SSO can be achieved with those SPs that choose to trust the Payment System top-level CA corresponding to the cardholder’s card. From the SP perspective, authentication/SSO can be facilitated only for those cardholders whose Payment System top level CA a given SP has chosen to trust. The architecture does not provide for explicit trust management at the Issuer level. This feature is inherited from the EMV PKI, which does not allow merchant terminals *not* to trust individual Issuers that have been certified by a trusted CA. This arises from the fact that EMV was designed for use within a closed environment in which all parties (Issuers and Acquirers) have signed an agreement with the brand. Indeed, even in the non-electronic world, merchants are typically required to accept all cards bearing the brand as part of the condition of being an approved merchant.

3.3 The SSO Protocol

This section describes the protocol of the SSO scheme. The protocol starts when the user requests a protected resource from the SP. As we see below, the same protocol can also be used for initial user registration at an SP.

The protocol assumes that the SP has already been authenticated to the user (cardholder) by some mechanism outside the scope of this paper. Specifically, it is assumed that, as part of that mechanism, the user manually verifies the SP’s unique identifier and, ideally, a cryptographically protected session is set up between the SP and the CS. A suitable mechanism for SP authentication is, for example, an SSL/TLS channel with server-side certificates⁶ [6]. In this case the SP’s unique identifier, the SPID, would be a field in its SSL/TLS certificate (typically its unique URL). Here it is worth noting that, if SSL/TLS is used for SP-to-user authentication, our scheme essentially uses the commercial PKI established on the Internet for SSL/TLS connections to facilitate SP-to-user authentication, and the EMV PKI established for credit/debit card payments to facilitate user-to-SP authentication⁷. A detailed description of the SSO protocol follows.

1. The SP sends an authentication request message to the CS. This message contains a freshly generated nonce and an indication saying whether or not PIN verification is required.

⁶ Since the user requests a protected resource, it is likely that an SSL/TLS connection will be required anyway.

⁷ From this perspective, the paper could be re-titled ‘Integrating EMV certificates and SSL’. However, SP-to-user authentication schemes other than SSL could also be used.

2. The CS selects the card's AA, initiates application processing and performs processing restrictions, as explained in steps 1-3 in section 2. If this step fails, SSO also fails.
3. If PIN verification was required in step 1, the CS performs offline PIN verification with the card, as explained in section 2.2.
4. The CS performs DDA with the card. The main difference from the 'standard' DDA (as explained in section 2.1) is that the nonce used with the INTERNAL AUTHENTICATE command is the SP-provided nonce from step 1. The SP's Identifier (acquired during SP-user authentication, as explained above) is also included in the data passed to the card for signing. It should be noted that the CS cannot verify the card's and the Issuer's public key certificates as it does not have the root CA's public key.
5. The CS sends an authentication assertion message back to the SP. This message includes the following data structures obtained from step 4.
 - The card's anonymous certificate.
 - The card Issuer's public key certificate.
 - The card's signature produced as part of DDA. This signature covers the nonce of step 1, the SPID and the PVDE, as explained in section 5.
 - Any other data that is input to the card signature calculation.
6. The SP verifies the Issuer and card public key certificates, as explained in section 2.1. The SP also makes sure that the card has not been blacklisted and that the aforementioned certificates have not been revoked. If this step fails, SSO also fails.
7. The SP reconstructs the data structure that was signed by the card in step 5 and verifies the signature using the card's public key. If verification is unsuccessful, SSO fails.
8. The SP assesses the data used to compute the card's signature. In particular, the SP checks the SPID and makes sure that it indeed represents this SP (and not any other). Furthermore, the SP assesses the PVDE. If the SP's requirements are met, SSO succeeds and access to the protected resource is granted. Otherwise, SSO fails.

The CS's response (step 5) does not contain any personally identifying information about the cardholder. The SP may, however, differentiate between users based on the unique (Serial Number, Issuer Identifier) pair included in the card's anonymous certificate. Furthermore, the protocol can be used for initial registration of a user at a SP; the SP creates a new user account for a newly encountered anonymous certificate.

The CS can achieve SSO at disparate SPs by running the protocol whenever needed. Of course, the card needs to be in the card reader of the CS during the protocol run. If PIN verification has been performed, the card needs to remain in the reader between protocol runs so that the session state is maintained within the card.

4 Threat Analysis

In this section threats to the scheme are evaluated.

4.1 SP Collusion

If a number of SPs collude, they can trivially compromise user privacy by correlating the unique identifying (Serial Number, Issuer Identifier) pairs found in the card anonymous certificates. The scheme does not address this threat.

However, as also pointed out in [7], complete prevention of a ‘SP collusion’ attack is difficult as SPs may also be able to correlate users based on other profile information they may maintain (such as names or telephone numbers). As stated in the Liberty specifications [7, p.71], ‘The only protection is for Principals [users] to be cautious when they choose service providers and understand their privacy policies’.

4.2 Reflection Attack

An attacker could forward the authentication request message (step 1 of the protocol) received from an SP as part of the SSO process to a victim user, while masquerading as the SP to that user (maybe by spoofing the SP’s interface and SPID). Forwarding the user’s valid response (step 5) to the SP might result in successful impersonation.

This attack is prevented as long as the SP authentication method is secure, it results in a cryptographically protected session and it is correctly performed by the user. In the case of SSL/TLS this involves the user inspecting the SP’s URL and making sure that it indeed represents the desired SP.

As the CS’s authentication response contains the SP’s unique identifier, which is digitally signed by the card, intermediaries (such as an attacker) cannot change it without being detected. At the same time the SP is given assurance that the response is indeed meant for this particular SP (and not any other).

It should be noted that the attack is *not* prevented if launched by a dishonest SP. As explained in section 4.1, users should be cautious when they choose SPs.

4.3 Traffic Analysis

An attacker capable of monitoring network traffic between the CS and SPs could compromise the user’s privacy in that the attacker will learn which SPs the user is communicating with. The attack cannot be prevented by encrypting traffic (using SSL/TLS, for example), as packet headers typically need to be available in the clear for routing purposes. This threat is outside the scope of the scheme described here, but could be addressed separately using anonymising techniques, such as those described by Chaum [8].

4.4 Attacks Using a Malicious Cardholder System

The EMV specifications make no provision for cards to authenticate merchant terminals prior to releasing information. As a result, when the card is inserted into the CS, it may be possible for malicious software in the CS to extract private information (such as the cardholder’s Primary Account Number which is

likely to be stored in the card) and to disclose it to unintended parties. Similarly, if the card reader does not have its own (trusted) PIN pad, the CS could collect cardholders' AA PINs⁸. Furthermore, a malicious CS could spoof local and remote user interfaces and abuse the user's authentication status at SPs by modifying traffic or hijacking the entire session, even if communications are 'cryptographically protected'. Thus, the SSO agent has to be trusted by the user not to engage in such behaviour and its integrity has to be protected. Having it signed by a party trusted by the user (e.g. the card Issuer) might address the threat, but risks remain if other malicious software is executed on the CS.

However, despite these threats, the scheme provides two-factor user authentication (proof-of-possession of the card and, if required, proof-of-knowledge of the PIN), even in the presence of a malicious CS: firstly, it requires the EMV card to be present in the CS; without it user authentication (and therefore illegitimate impersonation) will fail. Secondly, the scheme protects against the CS falsely pretending that PIN verification took place; the PIN verification status maintenance is managed by the trusted card itself, as explained in section 5. This protects against PIN compromise by a malicious CS as the PIN never needs to be inserted into the device (for SPs that do not require PIN verification).

4.5 Stolen EMV Card

Stolen EMV cards allow attackers to impersonate users to SPs that do not require PIN verification. The obvious countermeasure is for SPs to require PIN verification. In this case the attacker will not be able to impersonate the legitimate cardholder, even by using a maliciously modified CS. Of course, if the attacker also has access to the user's PIN, then impersonation will be successful.

In order to guard against 'stolen card' attacks, SPs should follow the same procedures as merchant terminals. In particular they should periodically contact card Issuers and/or Payment System CAs to obtain Certificate Revocation Lists (CRLs) and/or blacklisted card information. Step 6 in of the SSO protocol (section 3.3) provides for checking of these CRLs and blacklisted card information.

4.6 Service Denial Attacks

The scheme requires the SPs to check whether the signature returned by the CS is computed using the correct nonce, i.e. it requires the SP to maintain state while waiting for the response from the CS. This potentially opens the door to service denial attacks. However, we have assumed prior SP-to-user authentication, which ideally results in a secure session. This means that, before the SSO protocol is executed, the SP has established that it is talking to an existing client for whom it has already created some state. The fact that the SP is required to remember a nonce for each user-to-SP authentication attempt, is thus not likely to significantly increase the SP's exposure to service denial attacks.

⁸ The PIN used by the AA should be separate from the PIN(s) used by EMV applications that may coexist on the card.

4.7 Signature Oracle Attacks

The SSO scheme, as described in sections 3.1 and 3.3 (step 5), involves the card signing a data string containing a nonce supplied by the SP. Thus the protocol involves the card signing a message, part of which is provided by an external party. There exists the possibility that this could be used as part of an ‘oracle attack’, where the card is persuaded to sign a string that could be used in another application using the same key pair. The reason why this is not a significant threat in the case of the EMV payment application is that the signed string is different in format from the one expected by an EMV application.

5 Advantages and Disadvantages

This section discusses the advantages and disadvantages of the described SSO scheme.

5.1 Advantages

Advantages of the authentication/SSO scheme described in this paper include the following.

- The scheme reuses the existing EMV PKI which is already established on a world wide basis.
- The scheme does not require a continuous online presence of the card Issuer.
- Once the authentication/SSO protocol has completed successfully, subsequent protocol runs do not necessarily require user intervention. This yields transparent user authentication at subsequently used SPs.
- As user authentication may be transparent, the protocol can be repeated whenever appropriate. An online banking SP, for example, may wish to ensure that the cardholder’s card is still present in the CS whenever access to a sensitive resource is requested. Rerunning the SSO protocol during a session increases the achieved level of security without usability implications.
- No identifying information about the user is included in the messages exchanged. This protects the user’s anonymity and privacy. Furthermore, no risks of personal information exposure arise at the SP.
- Maliciously acting devices can only compromise the user’s current session or impersonate users while the EMV card is present. Furthermore, they cannot falsely pretend that PIN verification took place successfully.
- The scheme does not necessarily require an online third party. SPs need, however, to follow the same principles and policies as merchant terminals with respect to the certificates used.
- The scheme preserves user mobility.
- The scheme can potentially be adapted as a new Liberty Alliance [7] profile.

5.2 Disadvantages

Disadvantages of the authentication/SSO scheme described in this paper include the following.

- Issuers must install a separate EMV application on the card in order to support user authentication. This is a potentially significant cost. This cost is minimised if the AA is installed on the card at the time of issue, and it has to be weighed against the potential benefits gained by the Issuer. These might include new revenue streams from SPs that benefit from the AA.
- The cards used must be DDA-capable. The cost of DDA-capable cards is higher than the cost of cards not capable of DDA.
- It obviously works only for EMV cardholders equipped with card readers. The cost of the card reader (and maintaining the SSO agent), has to be weighed against the convenience offered by SSO.

6 Related Work

Single sign-on architectures within enterprise environments are examined in [9]. Currently deployed or proposed SSO schemes for open environments are based on a continually online ASP [10, 11, 12]. The scheme proposed in this paper, on the other hand, does not necessarily require the continuous online presence of any party; it falls into the category of *local* true SSO schemes [13]. Being based on a different trust model, it also constitutes an interesting alternative to the aforementioned schemes.

Other related work includes [14], where a security-enhanced e-commerce transaction scheme based on EMV cards is proposed. The scheme makes use of DDA and offline PIN verification in order to facilitate card and cardholder authentication respectively. Being a payment scheme, however, it requires online presence of the Issuer and does not aim for user privacy protection.

An annex of [3] describes how to combine the Secure Electronic Transaction (SET) protocol⁹ with EMV-compliant cards for electronic transactions conducted over the Internet. The complexity of the scheme is quite high. In addition, it requires the online presence of a ‘Payment Gateway’ which is connected to the Acquirer’s (and Issuer’s) legacy network.

Finally, it is worth noting that Subscriber Identity Module (SIM) cards of mobile phones have recently been augmented with EMV-compliant applications (<http://www.oberthurcs.com>) and that mobile equipment with EMV-compliant card readers has been available for some time.

7 Conclusion

In this paper we have proposed a SSO scheme which relies on EMV-compliant cards for cardholder authentication at SPs. These cards need to be able to per-

⁹ <http://www.setco.org>

form asymmetric cryptographic functions (i.e. DDA) and must have a separate EMV ‘Authentication Application’ installed on them by Issuers.

The CS itself acts as ASP for relying SPs. The scheme does not require on-line participation of the Issuer and its security does not depend on CS integrity, as core functions are delegated to the trusted card. It leverages the existing and established EMV PKI and preserves user mobility and privacy, and can be regarded as an alternative to other smartcard-based user authentication mechanisms (such as Subscriber Identity Modules).

The associated SSO protocol only requires minimal interaction, yielding a potentially seamless user experience and allowing several transparent re-authentications to occur within a given user/SP session.

References

- [1] EMV. *EMV2000 Integrated Circuit Card Specification for Payment Systems Version 4.0 — Book 1: Application Independent ICC to Terminal Interface Requirements*, December 2000.
- [2] EMV. *EMV2000 Integrated Circuit Card Specification for Payment Systems Version 4.0 — Book 2: Security and Key Management*, December 2000.
- [3] EMV. *EMV2000 Integrated Circuit Card Specification for Payment Systems Version 4.0 — Book 3: Application Specification*, December 2000.
- [4] EMV. *EMV2000 Integrated Circuit Card Specification for Payment Systems Version 4.0 — Book 4: Cardholder, Attendant and Acquirer Interface Requirements*, December 2000.
- [5] Cristian Radu. *Implementing Electronic Card Payment Systems*. Computer Security Series. Artech House, Norwood, 2002.
- [6] Eric Rescorla. *SSL and TLS*. Addison-Wesley, Reading, Massachusetts, 2001.
- [7] Liberty Alliance. *Liberty ID-FF Bindings and Profiles Specification version 1.2-08*, April 2003.
- [8] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.
- [9] Jan De Clercq. Single sign-on architectures. In George I. Davida, Yair Frankel, and Owen Rees, editors, *Proceedings of the Infrastructure Security Conference (InfraSec 2002)*, volume 2437 of *Lecture Notes in Computer Science*, pages 40–58. Springer-Verlag, Berlin, 2002.
- [10] Liberty Alliance. *Liberty Architecture Overview*, November 2002.
- [11] OASIS, <http://www.oasis-open.org/committees/security/>. *Security Services Technical Committee Homepage*.
- [12] Andreas Pashalidis and Chris J. Mitchell. A taxonomy of single sign-on systems. In R. Safavi-Naini and J. Seberry, editors, *Information Security and Privacy, 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11, 2003, Proceedings*, volume 2727 of *Lecture Notes in Computer Science*, pages 249–264. Springer-Verlag, Berlin, July 2003.
- [13] Andreas Pashalidis and Chris J. Mitchell. Single sign-on using trusted platforms. In C. Boyd and W. Mao, editors, *Information Security, 6th International Conference, ISC 2003, Bristol, UK, October 2003, Proceedings*, volume 2851 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, Berlin, 2003.

- [14] V. Khu-smith and C.J. Mitchell. Using EMV cards to protect e-commerce transactions. In A. Min Tjoa K. Bauknecht and G. Quirchmayr, editors, *Proceedings of the 3rd International Conference on Electronic Commerce and Web Technologies (EC-Web 2002)*, volume 2455 of *Lecture Notes in Computer Science*, pages 388–399. Springer-Verlag, Berlin, 2002.

Distributing Security-Mediated PKI^{*}

Gabriel Vanrenen and Sean Smith

Department of Computer Science/PKI Lab
Dartmouth College, Hanover NH 03755 USA
gabriel.vanrenen@alum.dartmouth.org
sws@cs.dartmouth.edu

Abstract. The SEM approach to PKI (by Boneh et al [4]) offers many advantages, such as instant revocation and compatibility with standard RSA tools. However, it has some disadvantages with regard to trust and scalability: each user depends on a mediator that may go down or become compromised.

In this paper, we present a design that addresses this problem. We use secure coprocessors linked with peer-to-peer networks, to create a network of trustworthy mediators, to improve availability. We use threshold cryptography to build a back-up and migration technique, to provide recovery from a mediator crashing while also avoiding having all mediators share all secrets. We then use strong forward secrecy with this migration, to mitigate the damage should a crashed mediator actually be compromised. We also discuss a prototype implementation of this design.

1 Introduction

In this paper, we apply tools including peer-to-peer computing and secure coprocessors to distribute the SEM approach to PKI, and thus preserve its advantages while overcoming its scalability, reliability, and trust problems. Sect. 2 reviews the SEM approach, and discusses its advantages and disadvantages. Sect. 3 discusses the tools we apply to this problem. Sect. 4 discusses the design we build with these tools. Sect. 5 discusses our prototype. Sect. 6 discusses some related approaches. Sect. 7 discusses some conclusions and future work.

2 SEM

Motivation In PKI, a *certificate* is a signed assertion binding a public key to certain properties. The correctness of the trust decisions a relying party makes depends on the assumption that the entity knowing the matching private key possesses those properties. When this binding ceases to hold, this certificate needs to be *revoked*, and this revocation information needs to propagate to relying parties, lest they make incorrect trust judgments regarding that public key.

^{*} This work was supported in part by the Mellon Foundation, by the NSF (CCR-0209144), by Internet2/AT&T, and by the Office for Domestic Preparedness, U.S. Dept of Homeland Security (2000-DT-CX-K001). The views and conclusions do not necessarily represent those of the sponsors.

Consequently, fast and scalable certificate revocation has been area of active research in recent years (e.g., [21, 23]). In their *Security Mediator*¹ (SEM) approach, Boneh et al [4] proposed a system that revokes the ability of the keyholder to use a private key, instead of (or in addition to) revoking the certificate attesting to the corresponding public key.

Architecture The SEM approach is based on *mediated RSA (mRSA)*, a variant of RSA which splits the private key of a user into two parts. As in standard RSA, each user has a public key (n_u, e_u) and a private key d_u , where n is the product of two large primes, $\gcd(e_u, \phi(n_u)) = 1$, and $d_u * e_u = 1 \pmod{\phi(n_u)}$. The public key of a user u is the same as in standard RSA, as is the public-key operation. The two parts of a user's secret key are $d_{\text{sem},u}$ and $d_{\text{user},u}$, where d_u is the standard secret key and $d_u = d_{\text{sem},u} + d_{\text{user},u} \pmod{\phi(n_u)}$. $d_{\text{user},u}$ is the part held by the user and $d_{\text{sem},u}$ is the part held by the SEM. (We note that $d_{\text{sem},u}$ and $d_{\text{user},u}$ are each statistically unique for each user u .)

This division of the secret key requires changes to the standard RSA key setup because a SEM must not know $d_{\text{user},u}$ and a user must not know $d_{\text{sem},u}$. So, a trusted party (e.g., a CA) performs key setup by generating a statistically unique $\{p_u, q_u, e_u, d_u, d_{\text{sem},u}\}$ for a user u . The private key d_u is generated in the standard manner, but is communicated to neither the SEM nor the user. Instead, $d_{\text{sem},u}$ is chosen as a random integer in $[0, n_u - 1]$, and $d_{\text{user},u}$ is then calculated as $d_{\text{user},u} = d_u - d_{\text{sem},u} \pmod{\phi(n_u)}$.

Because the private key d_u is split into two “halves,” private key operations require the participation of both the user and the SEM: e.g., each party raises the message to its half-exponent, modulo n , and the results are then multiplied, also modulo n . (See Fig. 1.) Thus the full private key never needs to be reconstructed

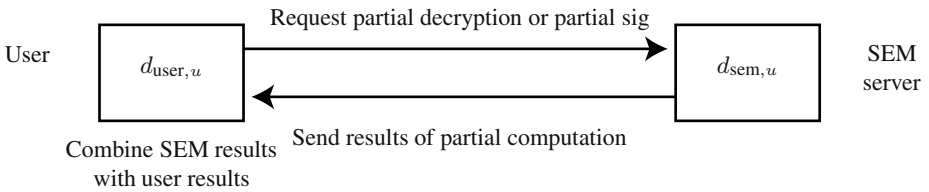


Fig. 1. The general SEM algorithm.

Advantages The SEM approach provides several advantages. Since these essentially are standard RSA operations, a SEM PKI is compatible with most legacy public-key cryptography tools. Since a full private-key operation can occur only if the SEM believes the user's key pair is valid, then the system can revoke a key pair by having the SEM refuse to carry out its half of the operation. This approach can reduce or even eliminate (in the case of revocation due to administrative action, such as a user ceasing employment) the need for certificate revocation lists—since a private-key operation (such as signature or decryption) cannot occur after revocation.

¹ Also referred to as “semi-trusted mediator.”

Furthermore, the SEM itself gains no useful information in servicing users. When decrypting, the SEM receives the ciphertext but is only able to partially decrypt it, so no useful information could be gained by a malicious SEM. For signature generation, a user sends the SEM a hash of the message which the SEM uses to generate the signature. This also contains no information about the cleartext of the message itself, so a user's data is kept confidential.

Additionally, the compromise of a single SEM does not compromise the secret keys of any users. Instead, the attacker is able to revoke the security capabilities for users connected to the SEM. Although Boneh et al state that attackers could unrevoke revoked certificates, this can be prevented by having honest SEMs permanently delete $d_{\text{sem},u}$ upon revocation.

Disadvantages However, the initial SEM approach has scalability disadvantages. In a large-scale distributed system, we must allow for problems such as mobile users, network partitioning, crashing of machines, and occasional compromise of servers. To accommodate a large population, we could add multiple SEMs. However, if a user's $d_{\text{sem},u}$ lives on exactly one SEM, then we have many problems: temporary denial of service if the network is partitioned; permanent denial of service if the SEM suffers a serious failure; inability to revoke the key pair if an adversary compromises a SEM and learns its secrets.

In their original paper, Boneh et al did propose one way to distribute the SEM architecture by using a stateless model in which a user can connect to any SEM. However, this approach required that the entire SEM network have a single RSA key pair, so that any node can access the encrypted $d_{\text{sem},u}$ bundled with each request. This network-wide key pair could either be stored on each island through replication or shared securely among islands using threshold cryptography. In the first case, compromise of a single island is potentially easier (due to replication) and causes damage to the entire network. In the latter case, each user request requires distributed computation among islands which hurts performance. In either case, the user is at risk if she connects to a compromised SEM.

3 Tools

To address the problem of distributing SEM, we use several tools.

We need to be able to trust a SEM to use and delete each user's $d_{\text{sem},u}$ when appropriate, and not transmit it further. However, the more we distribute the SEMs throughout a network, the less foundation we have for such trust. To address this problem, we use *secure coprocessors*, such as the IBM 4758 [26]. This gives us a general-purpose computing environment and cryptographic protections, coupled with high-assurance protection against physical attacks, and an *outbound authentication* scheme which lets software applications running on the coprocessor authenticate themselves to remote parties [25]. This platform thus gives us a safe and confidential environment in remote environments. If a user trusts our software is not flawed, then the user can also trust that software executed cannot be altered by adversaries and the user may also remotely authenticate instances of this software. (In Sect. 7, we consider using newer trusted computing platforms as well.)

We'd like to make it easy for users to find SEMs (and for SEMs to find each other), and we'd like this functionality to persist despite failures and (potentially) malicious attacks. *Peer-to-peer networking (P2P)* (embodied by technology such as Gnutella) is an attractive choice here. With P2P, communication does not rely on a central entity to forward requests or messages. Rather, each entity either tries to satisfy a request itself, or forwards it to its neighbors, in the spirit of the older distributed concept of *diffusing computation* [2].

This decentralization is a key benefit of the peer-to-peer network, as it removes any central entity necessary for the system to function. Without a central controlling server, the network's survivability increases by causing denial of service attacks to be much more difficult (as the RIAA has found to its dismay). Additionally, the damaging effects of network partitions are potentially alleviated by standard P2P communication algorithms, as a new path to a destination may be found.

We will also need to distribute critical secrets across multiple SEMs, for resilience against attack. Here, we can use the standard technique of *threshold cryptography* [24]. Given a secret y and parameters $t < k$, we construct a degree t polynomial that goes through the point $(0, y)$, and choose k points on this polynomial as *shares* of y . Any t shares suffices to reconstruct the polynomial and hence y , but fewer than t shares give no information.

We need to accommodate the fact that machines may be compromised, and the secrets they store may become exposed to the adversary. To mitigate the damage of such potential exposure, we can use the technique of *strong forward security* [5]. We divide time into a sequence of clock periods, and use a cryptographic system such that even if the private key for a given period is exposed, use of the private key in previous or future sessions is still secure. Burmester et al give two examples of strong forward secure schemes, one for any public key cryptosystem and another for use in an El Gamal key escrow system.

4 Design

Architecture In our basic architecture, we envision SEMs as trustworthy *islands* distributed throughout the network. We use a secure coprocessor to house each SEM and thus give it a foundation for this trustworthiness. As noted earlier, this technology also lets each island have a key pair and certificate chain that establishes the entity who knows the private key is an instantiation of our island software on an untampered device. Thus, users can authenticate islands, and islands can authenticate each other.

Each island will house resources that enable it carry out services. When a user requests such a service, we use P2P techniques to carry the request to the proper island, and carry the response back to the user. (As we discuss below, individual islands will also house resources other islands need; we can use P2P there as well.)

Despite physical protections, an individual island may still become compromised and reveal its data to the adversary. An individual island may also become unavailable, due to crash or partition. To handle these scenarios, we build a *migration* scheme based on threshold cryptography and strong forward security.

When we initially create a secret x and transmit it to an island L , we also split into k shares using threshold cryptography. We securely transmit each share of x to a

different island. (Additionally, the shares may be proactively updated using the techniques described in [11, 12, 14, 15] so that an attacker may not slowly acquire enough shares to reconstruct x .) After those steps are complete, the secret is stored both on the primary island L and on k other islands, so an attacker must either compromise L or compromise t of the k islands in order to get x .

When island L is unavailable to fulfill a request that requires x , then the requester will have to be redirected to another island M , and the shareholders will need to participate in reconstructing x there. However, since the original island L may have been compromised, x must be updated using strong forward security so that the old version on L is rendered useless.

The general migration scheme is executed as follows:

1. The user tries to connect to the assigned island L , but fails.
2. The user then connects to another island M instead. M may be chosen either pseudorandomly, using a load balancing algorithm or another scheme (e.g., based on network proximity to the user).
3. The islands that hold shares of x are contacted and, and this x is updated using strong forward security. As discussed below, this update may or may not involve reconstruction of x , depending on the method chosen. Generally, the strong forward security scheme will vary depending on the how the secret x is generated.
4. Strong forward security results in M storing the updated secret.
5. Migration is complete and M can then fulfill the user's request.

SEM Operations To use this architecture for SEM, each island acts as a SEM mediator, holding $d_{\text{sem},u}$ for a number of clients. We distribute load across the islands by, at key generation, assigning users to different SEMs. (We could also distribute load via migration.) As with the original SEM architecture, a user's $d_{\text{sem},u}$ is stored in full only on one island.

In the original SEM scheme [4], a CA generates key pairs for users and splits d into two halves. In our variant, the CA must additionally share $d_{\text{sem},u}$ to k islands in the network using threshold cryptography. (See Fig. 2 Also stored with those shares is the user's identity and the revocation status of the user's key pair (initialized to "false," not revoked). For key generation, the CA must be able to prove its identity to the islands; otherwise, the islands will ignore its request. If we desired an escrow service to allow authorized decryption of data after revocation (or if we do not decide to use the CA during migration, as discussed below), we also distribute shares of the full secret key d_u .

If the island that holds $d_{\text{sem},u}$ and revocation information for a user u goes down, then the other islands must be able to determine whether the user's key pair has been revoked. We accomplish this by, during revocation, having the shareholders as well as original island update the revocation status for that key pair. In our initial vision, we delete the shares of $d_{\text{sem},u}$ that are stored on k other islands.

Assume that a user's $d_{\text{sem},u}$ is stored on island L . The network is notified that a user's key pair is to be revoked, and a P2P request is generated to L to revoke the user's key pair. If L is operational, then L notifies all of the other islands that hold shares of $d_{\text{sem},u}$ to delete them and store the fact that $d_{\text{sem},u}$ has been revoked; L also deletes

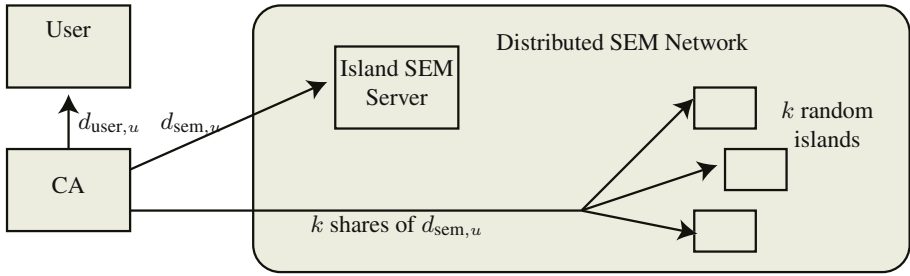


Fig. 2. Key set-up in our proposed system

$d_{sem,u}$ and adds the user's serial number to its CRL. Else if L is not operational, then the k islands holding the shares of $d_{sem,u}$ are notified and told to delete their shares. (Note that in this case, migration—see below—has not yet occurred for this user or else an island would have been contacted.)

SEM Migration If a user u issues a request but the island L holding $d_{sem,u}$ is not available, then we select another island M and request migration. As with other selections, M may be chosen in a number of ways (although a random or pseudorandom way, so that an attacker cannot predict it, would help in some scenarios—we plan to add this in future work). See Fig. 3.

After that initial step is performed, we have two different approaches, depending on whether a CA exists that can know the full private key d_u . Any communication between the islands is authenticated using the outbound authentication of the secure coprocessors and it is assumed that the online CA also has some mode of outbound authentication to prove the source of its messages.

For added resilience, we can have shareholder islands not participate in migration if they can still ping the original island L .

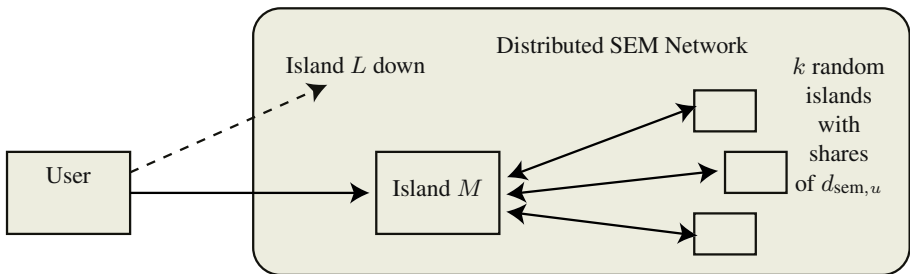


Fig. 3. Migration in our proposed system

If we have an online CA, the new island M contacts it and tells it that migration is to occur. (If we constrain the choice of the next M , then the CA must verify that

M is a satisfactory candidate.) M sends a request for any islands in the network that have shares of $d_{\text{sem},u}$ to send those securely to the CA using standard RSA encryption. The CA can then reconstruct $d_{\text{sem},u}$. If the CA stores the full private key for d_u , then it can use that. Otherwise, the shareholders for d_u must also send those, so the CA can reconstruct it as well. The CA generates a random number x in the interval $[0, n_u - 1]$ such that $x \neq d_{\text{sem},u}$, and calculates $y = d_u - x \pmod{\phi(n_u)}$. The CA securely distributes shares of x to the k shareholder islands using threshold cryptography; the shareholder islands delete the old shares for $d_{\text{sem},u}$. The CA securely sends y to the user by encrypting it with the user's public key and then partially decrypting with the old $d_{\text{sem},u}$. The user deletes the old $d_{\text{user},u}$ and sets $d_{\text{user},u} = y$. The CA securely sends the x to M , who deletes the old $d_{\text{sem},u}$ and sets $d_{\text{sem},u} = x$.

Once the user receives y , it reconnects to the network and performs the operation again, using its new value $d_{\text{user},u}$. At this point, M can complete the request and the migration is complete.

If no CA exists, then we need to generate a new $d_{\text{sem},u}, d_{\text{user},u}$ pair without reconstructing d_u or $\phi(n_u)$, since we do not have a safe place to store them.

For now, we borrow the trick of [16] and have M generate a δ in a range $[-r, r]$, and changing $d_{\text{sem},u}$ to $d_{\text{sem},u} - \delta$, where r is big enough to keep the key halves changing unpredictably, but small enough to be smaller than $d_{\text{sem},u}$ and $d_{\text{user},u}$ for a practically indefinite number of rounds. M sends δ to the user (encrypted with u 's public key and then partially decrypted with the old $d_{\text{sem},u}$); the user replaces $d_{\text{user},u}$ with $d_{\text{user},u} + \delta$. (We could also use the [16] trick of having M and u together pick r , to reduce risk from a compromised M .)

This way, neither M nor u need to know $\phi(n_u)$, but the new $d_{\text{user},u}$ and $d_{\text{sem},u}$ remain positive and still sum to d_u . M splits r into k shares and sends each to a $d_{\text{sem},u}$ shareholder; each shareholder uses its piece to to update its share.

It is tempting to have M pick a new $d_{\text{sem},u}$ directly and distribute shares to the shareholders of d_u , who then calculate the new $d_{\text{user},u}$ in a nicely distributed fashion. However, as of this writing, we cannot see how to reduce $d_u - d_{\text{sem},u}$ to $d_u - d_{\text{sem},u} \pmod{\phi(n_u)}$ without reconstructing $\phi(n_u)$.

Once the user receives the new $d_{\text{user},u}$, it can compute its half of the normal computation using the new $d_{\text{user},u}$. At this point, M can also complete its half of the computation because it has generated a new $d_{\text{sem},u}$ and the migration is complete.

As an area of future work, we are also considering incorporating strong forward secrecy into regeneration of the user's private key during regeneration of $d_{\text{sem},u}$. We already have trusted hardware, one of the components of some SFS schemes in the literature (e.g., [32, 9]). Furthermore, this would protect against compromise of L by the user u , in order to obtain $d_{\text{sem},u}$ and reconstructing d_u .

When an island goes down (or is compromised and subsequently shut down and restarted from a clean state), it has a few options upon reboot:

The island could delete all of the key halves it has stored, and thereby force users to migrate back to it. New users would also be assigned to it. It also deletes all of the shares of that it stored and requests new shares of those to be generated.

Alternatively, the island could poll the other islands to determine which $d_{\text{sem},u}$ halves have migrated away from it. It then deletes the information for the users that

migrated away and continues serving the other users. The island can continue using the shares it has stored, but it must determine whether any of them are out of date. Since the $d_{\text{sem},u}$ shares must be updated during migration, the $d_{\text{sem},u}$ shares could be invalid and the network must be polled to determine whether this is the case. If so, then the outdated shares must be updated.

Analysis First, we consider compromise of specific entities.

If the CA has been compromised, then we have a serious problem, since the CA generates the users' initial key pairs, and in the CA-migration case, learns the new key pairs as well.

Alternatively, suppose L has been compromised. The island L holds $d_{\text{sem},u}$ for some number of users. Additionally, L stores shares for some other users in the distributed SEM network. We must assume that the attacker has access to all of these values, so now we analyze what privileges are granted by illicit access to them.

- If the attacker acquires $d_{\text{sem},u}$ for another user, then migration effectively disables this $d_{\text{sem},u}$ because it causes a new $d_{\text{user},u}$ to be issued to the user. (Also, note that the new $d_{\text{user},u}$ is not sent back to L .) Since the new $d_{\text{user},u}$ does not mesh with the old $d_{\text{sem},u}$ due to the mRSA protocols, the old $d_{\text{sem},u}$ is rendered useless.
- If the attacker acquires $d_{\text{sem},u}$ and colludes with that user, then the attacker will be able to compute d_u from $d_{\text{user},u}$ and $d_{\text{sem},u}$, so migration fails to achieve full security in this case (unless, as discussed earlier, we try implementing SFS here as well).
- If the attacker colludes with a user whose key-half is not on that island, then the user and attacker might trick the SEM network to migrating that user's data to L , and thus reconstruct $d_{\text{sem},u}$. The user will then be able to reconstruct d_u , the full private key. This problem can be mitigated by using pings (as stated in section 4.3) to ensure that the user's main island is unavailable, and also using a non-predictable way to generate the next island for migration (to reduce the chance that M is a valid candidate). However, such problems may be the inevitable cost of higher availability in the distributed SEM network.
- If the attacker acquires shares of a $d_{\text{sem},u}$, the attacker effectively acquires no valuable information, unless the attacker also gains access to enough other shares of either in order to reconstruct them. However, this can be made extremely difficult using proactive share updating, as described in [11, 12, 14, 15].

Clearly, there is a period between the time of compromise time when that compromise is discovered. However, the use of secure coprocessors causes any physical attacks to be detected immediately (with high assurance) and stopped by the zeroization of coprocessor data.

If M is compromised, then the attacker gets access to the new $d_{\text{sem},u}$, so the migration is unsuccessful. However, as long as another migration to an uncompromised island can be performed, the $d_{\text{sem},u}$ acquired by the attacker can be rendered useless as described above. Additionally, in this case the attacker could send the user fake data for the new $d_{\text{user},u}$, but any resulting inconsistencies with decryption or signature generation would just flag M as compromised. As stated above, with secure coprocessors the window of time before this compromise is discovered should be small.

Network Trust Model The primary parties that require use of the network are the islands that comprise the network itself. Each island must have exclusive access to certain services in order to provide fast revocation of security capabilities. However, the CA (and users and islands) must also be able to gain access to the network services in a restricted way. Clearly, the requests of each party will differ and there must be a clear delineation of capabilities between them. For example, during migration, islands will have to search the network to find other islands that hold the shares of a user's $d_{\text{sem},u}$. Although this operation can be executed by the CA as well (when CAs can perform migration), users should not be able to (easily) determine the location of or acquire shares.

Islands join the network normally and become full members of it. Since each island in the network has a secure coprocessor with outbound authentication/attestation, each member in the peer-to-peer network can prove that it is a trusted island with certain privileges. This creates a trust network in which each island is known to be executing unmolested as long as our software is not flawed (and the coprocessor's physical security protection works). Once an island has authenticated itself to the system, it can search the network, advertise services, and perform any other command allowed by the peer-to-peer software.

Certificate Authorities can interact with the network in one of two ways: (1) they can connect to an island server that provides an interface to the rest of the network; or (2) they can connect directly to the P2P network, but with limited capabilities (registration and, if implemented using a CA, then migration). For example, the CA must be able to somehow query the network during key generation to determine to which island to assign the user.

Users do not connect directly to the P2P network, but instead communicate with an island that provides indirect access to the services available on the network. For example, during normal operation, users connect directly to their assigned island, but if that fails, then the user must notify the P2P network that migration is necessary. Users do so by connecting to another island (available in a public list) and requesting the migration service. The user is then assigned to an island and further communication occurs directly between that island and the user.

5 Prototype

We are currently building a prototype of our Distributed SEM approach; at the time of this writing, our current code (2000 lines of Java) deals with the peer-to-peer aspects of key generation and execution of migration (See Fig. 4.)

The *Island Server Code* performs migration and the island's part of decryption and key and signature generation using $d_{\text{sem},u}$. This part also introduces the networking support for the islands. It is a combination of both the original SEM server code, along with our server-related migration code.

The code on each island is divided into two parts. The *P2P network code* consists of the peer-to-peer access layer and the protocols necessary for island communication. We use the Project JXTA open source framework [29] to accomplish this. Each island runs a server that accepts connections from only the SEM server on the same machine. The server application is built on top of JXTA and uses the peer-to-peer protocols, such

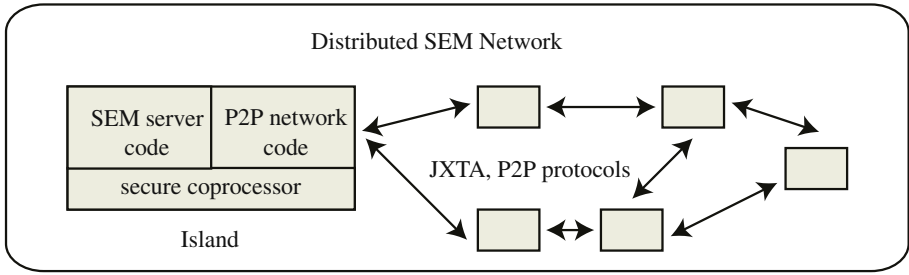


Fig. 4. Architecture of our system

as searching and pinging, available. It executes the distributed aspects of key generation (distribution of shares of $d_{\text{sem},u}$) and migration.

Specifically, during the distributed SEM operations we use JXTA's discovery service to find islands with the information needed (e.g. shares of $d_{\text{sem},u}$). We plan to leverage the security features of the JXTA framework to secure the P2P activities of the distributed SEM network. These include secure P2P groups to restrict access to the network and secure pipes to allow safe distribution of shares to an island during migration. Furthermore, we envision the use of outbound authentication data of secure coprocessors [25] in JXTA XML messages to validate the source of messages generated in the migration and revocation algorithms. The capabilities of the secure coprocessors will be exposed in the Java code using the Java Native Interface (JNI) [27].

The *SEM server code* accepts requests from users and handles most of those without using the P2P layer. When a migration or key registration request is received, however, the server code forwards the request to the internal server running JXTA and the internal server completes the request.

This is a modular approach that allows us to change the peer-to-peer implementation in the future. In the current prototype, we have implemented our network code in a simplified version. (Integration with the SEM server code, utilization of security features in the P2P layer, and porting on to the 4758 remain to be done.)

The *Certificate Authority Code* participates in key generation, as in the original SEM architecture. With our additions it is necessary for the CA to connect to an island and initiate a P2P registration request. We have implemented the code to perform registration in the network. (However, it is not yet integrated with the original SEM key generation code.)

The *Client User Code* combines the functionality of the original SEM architecture, which consisted of the user's half of signature generation and decryption, along with the additional steps required to request migration and process the migration response. (This remains a task for the future.)

Our code will be available for public download. Since the original SEM code is covered under the GPL, our changes—for migration and support of distributed functionality—are as well.

6 Related Work

Trusted Hardware and P2P Marchesini and Smith [18] built a *hardened Gnutella* P2P communication system within secure coprocessors; Boneh et al [13] also considered the potential of combining trusted computing with P2P.

Strong Forward Secrecy Tzeng and Tzeng [32] considered strong forward security with threshold cryptography for El Gamal signatures.

Standard Revocation Techniques *Certificate Revocation Lists* and variations on this method (e.g., Δ -CRLs) are one of the most common methods of certificate revocation. To revoke a certificate, the CA adds the serial number of the revoked certificate to the CRL and then publishes that CRL to a directory. Since this is only done periodically, CRLs are not a guarantee that a certificate is still valid. Also, since CRLs may be very large, users will generally not want to have to download them very often. In order to check whether a certificate is revoked, a user must potentially download a long CRL. To mitigate this problem, Δ -CRLs only distribute a list of the certificates revoked since the last CRL was distributed. Additionally, Cooper notes that when a new CRL is issued there will be a peak time in which many requests are made to download the CRL from the directory (because everyone wants to make sure that certificates aren't revoked and a CRL expires at the same time for everyone). He suggests spreading out the requests for CRLs over time by "over-issuing" CRLs such that a new CRL is published before the old one expires [7].

Another similar technique is *Windowed Key Revocation*, which uses CRLs but with a twist that certificates are assumed to be valid for a certain "window" of time and that CRLs have a reduced size due to the revocation window [20]. Additionally, in this scheme verifiers can control the allowed "window" time and to check if a certificate is revoked, the verifier checks the windowed CRL issued or grabs a new certificate from the CA.

The *Online Certificate Status Protocol (OCSP)* provides online verification that a certificate is still valid. This requires a CA to generate a digital signature for each request because the response from the CA must be signed [21]. The CA stores an internal log of the status of all certificates or possibly just a CRL that it doesn't publish. So, addition of revoked certificates is quick and the certificate status is updated instantly. A user must be online and must connect to the CA and check the status of a certificate.

Certificate status verification is a computationally expensive operation, as the response from the CA must be digitally signed. If a single validation server performs OCSP, then all requests must be routed to it, potentially overloading the server. Security may be weakened by a distributed environment because if any keys of any OCSP servers are compromised, then the entire system is compromised.

With *Certificate Revocation Trees*, instead of keeping entire list of revoked serial numbers, we keep a list of ranges of serial numbers that are good or bad. This saves space (better than standard CRLs), but adding a serial can involve a good amount of computation as it can require the entire tree to be recomputed. Still, revocation status can be quickly determined by a fast search through the tree.

Newer Revocation Techniques In [4], Boneh et al give a discussion of the benefits of the original SEM architecture with regards to other current solutions. Since the publication of that paper, a few other techniques for certificate revocation have been developed. Micali's *NOVOMODO* approach [23] uses one-way hashing and hash chains to show the validity or revocation status of certificates. Centralized *NOVOMODO*—in which the central secure server responds to all validity requests—is prone to performance issues and denial of service attacks as it is the central source for certificate validity proofs. Micali also presented a distributed version: having one central trusted server send out an array of the current validity proofs for all users to each server in the network. Micali does not discuss solutions to many potential problems that could occur during a distribution, such as bandwidth problems, network partition or untrusted server corruption. Micali states that it should be very difficult to attack the central server, as it does not accept incoming requests, but an attacker could instead attack the network surrounding the server, preventing it from distributing the array. In other words, distributed *NOVOMODO* still has a central “head” that can be severed (albeit in a more difficult way) in order to shutdown the system.

Ding et al introduce *Server-Aided Signatures (SAS)*, [8] a technique based on mediated cryptography similar to the SEM architecture with the focus on minimizing client computation load. While Distributed SEM works with both signature generation and decryption, SAS only deals with signature generation. It achieves a performance boost for the user by only requiring the user to compute a hash chain during setup, in a similar fashion to *NOVOMODO*, but differing in that the user keeps the chain secret. In SAS, the server must be stateful and, for each user, must save their certificate, i , and all of the signatures already generated for that user. This amount of state makes migration infeasible as every signature would have to be distributed on other islands using threshold cryptography. Additionally, in SAS the corruption of a server allows the attacker to produce user signatures because all of the prior signatures are saved on the server.

SEM Tsudik [31] and Boneh et al [3] have also followed up on their original SEM work. [16] explores adding proactive updates to the key halves.

7 Conclusions and Future Work

In this paper we have introduced a method to distribute SEM by using a network that combines the benefits of secure coprocessors and peer-to-peer networking, and provides providing efficient and uninterrupted access to private data stored on a trusted third party, even in the event of occasional server compromise. This approach avoids replication of data across the network while also avoiding the common use of distributed computation in order to access the secrets stored.

Once the implementation is completed, an area of great interest will be performance testing and tuning. The performance of both migration itself and the entire application running on the full P2P network (using secure coprocessors) will be reveal much information about our approach to distributed SEM—and the feasibility of this P2P/trusted hardware network.

Also, the IBM 4758 is a relatively expensive special-purpose device. Recent advances in *trusted computing*—both with COTS hardware (e.g., [10, 19, 30]) as well

with experimental CPUs (e.g., [6, 17, 22, 28])—explore the potential of achieving similar functionality (albeit a lower level of physical security) in standard desktop platforms. We plan to explore the potential (and relative performance) of distributed SEM on these platforms as well. We also plan to use our framework of P2P on trusted hardware to explore other applications as well. Finally, general Byzantine attacks must be considered in the Distributed SEM network and extra steps (e.g., [1]) must be taken to ensure the correct completion of all operations.

References

- [1] Alon, N., Kaplan, H., Krivelevich, M., Malkhi, D., Stern, J.: Scalable Secure Storage When Half the System Is Faulty. *Information and Computation* **174** (2002) 203–213
- [2] Andrews, G.: Paradigms for Process Interaction in Distributed Programs. *ACM Computing Surveys* **23** (1991) 49–90
- [3] Boneh, D., Ding, X., Tsudik, G.: Fine-Grained Control of Security Capabilities. *ACM Transactions on Internet Technology* (2004)
- [4] Boneh, D., Ding, X., Tsudik, G., Wong, C.M.: A method for fast revocation of public key certificates and security capabilities. In: 10th USENIX Security Symposium. (2001) 297–308
- [5] Burmester, M., Chrissikopoulos, V., Kotzanikolaou, P., Magkos, E.: Strong Forward Security. In: IFIP-SEC '01 Conference, Kluwer (2001) 109–121
- [6] Chen, B., Morris, R.: Certifying Program Execution with Secure Processors. In: 9th Hot Topics in Operating Systems (HOTOS-IX). (2003)
- [7] Cooper, D.A.: A model of certificate revocation. In: Fifteenth Annual Computer Security Applications Conference. (1999) 256–264
- [8] Ding, X., Mazzocchi, D., Tsudik, G.: Experimenting with server-aided signatures. In: Network and Distributed Systems Security Symposium. (2002)
- [9] Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong Key-Insulated Public-Key Schemes. In: Public Key Cryptography—PKC 2003, Springer-Verlag LNCS 2567 (2003) 109–121
- [10] England, P., Lampson, B., Manferdelli, J., Peinado, M., Willman, B.: A Trusted Open Platform. *IEEE Computer* (2003) 55–62
- [11] Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Optimal resilience proactive public-key cryptosystems. In: IEEE Symposium on Foundations of Computer Science. (1997) 384–393
- [12] Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Proactive RSA. In: Advances in Cryptology—CRYPTO 97, Springer Verlag LNCS 1294 (1997) 440–454
- [13] Garfinkel, T., Rosenblum, M., Boneh, D.: Flexible OS Support and Applications for Trusted Computing. In: 9th Hot Topics in Operating Systems (HOTOS-IX). (2003)
- [14] Herzberg, A., Jakobsson, M., Jarecki, S., Krawczyk, H., Yung, M.: Proactive public key and signature systems. In: ACM Conference on Computer and Communications Security. (1997) 100–110
- [15] Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: Advanced in Cryptology—CRYPTO 95, Springer Verlag LNCS 963 (1995) 339–352
- [16] Le, Z., Smith, S.: Proactive mediated rsa. Manuscript, Department of Computer Science, Dartmouth College (2004)
- [17] Lie, D., Thekkath, C., Mitchell, M., Lincoln, P., Boneh, D., Mitchell, J., Horowitz, M.: Architectural Support for Copy and Tamper Resistant Software. In: Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems. (2000) 168–177

- [18] Marchesini, J., Smith, S.: Virtual Hierarchies: An Architecture for Building and Maintaining Efficient and Resilient Trust Chains. In: *Proceedings of the 7th Nordic Workshop on Secure IT Systems—NORDSEC 2002*, Karlstad University Studies (2002)
- [19] Marchesini, J., Smith, S., Wild, O., Macdonald, R.: *Experimenting with TCPA/TCG Hardware, Or: How I Learned to Stop Worrying and Love The Bear*. Computer Science Technical Report TR2003-476, Dartmouth College (2003)
- [20] McDaniel, P., Jamin, S.: Windowed certificate revocation. In: *2000 IEEE Symposium on Security and Privacy*. (2000) 1406–1414
- [21] McDaniel, P., Rubin, A.: A response to can we eliminate certificate revocation lists? In: *Financial Cryptography*. (2000)
- [22] McGregor, P., Lee, R.: *Virtual Secure Co-Processing on General-purpose Processors*. Technical Report CE-L2002-003, Princeton University (2002)
- [23] Micali, S.: Novomodo: Scalable certificate validation and simplified pki management. In: *1st Annual PKI Research Workshop*. (2002)
- [24] Shamir, A.: How to share a secret. In: *Communications of the ACM*. Volume 22. (1979) 612–613
- [25] Smith, S.: Outbound Authentication for Programmable Secure Coprocessors. In: *Computer Security—ESORICS 2002*, Springer-Verlag LNCS 2502 (2002) 72–89
- [26] Smith, S., Weingart, S.: Building a High-Performance, Programmable Secure Coprocessor. *Computer Networks* **31** (1999) 831–860
- [27] Stearns, B.: *Trail: Java Native Interface*. Sun Microsystems, Inc. (2004)
<http://java.sun.com/docs/books/tutorial/native1.1/>.
- [28] Suh, G., Clarke, D., Gassend, B., van Dijk, M., Devadas, S.: AEGIS: Architecture for Tamper-Evident and Tamper-Resistant processing. In: *Proceedings of the 17 International Conference on Supercomputing*. (2003) 160–171
- [29] Sun Microsystems, Inc.: *Project JXTA: Java Programmers Guide*. (2001)
<http://www.jxta.org>.
- [30] Trusted Computing Platform Alliance: *TCPA PC Specific Implementation Specification, Version 1.00*. <http://www.trustedcomputinggroup.org> (2001)
- [31] Tsudik, G.: Weak Forward Security in Mediated RSA. In: *Security in Computer Networks Conference*. (2002)
- [32] Tzeng, Z., Tzeng, W.: Robust Key-Evolving Public Key Encryption Schemes. *Cryptography Eprint Archive Report* <http://eprint.iacr.org/2001/009> (2001)

Distributed CA-based PKI for Mobile Ad Hoc Networks Using Elliptic Curve Cryptography

Charikleia Zouridaki¹, Brian L. Mark¹, Kris Gaj¹, Roshan K. Thomas²

¹ George Mason University, Electrical and Computer Engineering Dept., 4400 University Drive, Fairfax, VA, 22030, USA

{czourida, bmark, kgaj}@gmu.edu

² McAfee Research, Network Associates, Inc., 1145 Herndon Parkway, Suite 500, Herndon, VA, 20170, USA

RThomas@nai.com

Abstract. The implementation of a standard PKI in a mobile ad hoc network (MANET) is not practical for several reasons: (1) lack of a fixed infrastructure; (2) a centralized certification authority (CA) represents a single point of failure in the network; (3) the relative locations and logical assignments of nodes vary in time; (4) nodes often have limited transmission and computational power, storage, and battery life. We propose a practical distributed CA-based PKI scheme for MANETs based on Elliptic Curve Cryptography (ECC) that overcomes these challenges. In this scheme, a relatively small number of mobile CA servers provide distributed service for the mobile nodes. The key elements of our approach include the use of threshold cryptography, cluster-based key management with mobile CA servers, and ECC. We show that the proposed scheme is resistant to a wide range of security attacks and can scale easily to networks of large size.

Keywords: mobile ad hoc network, threshold cryptography, elliptic curve cryptography, cluster, scalability

1 Introduction

Providing security in MANETs is an inherently challenging problem due to the lack of a fixed infrastructure, the dynamically changing network topology, the limitations of the wireless channel, and the limited capabilities of the nodes. Since the nodes are mobile, they are particularly vulnerable to physical attacks from within and outside the network. The nodes are typically of small size and have limited computational, storage, and transmission power, as well as limited battery life. Such limitations place severe constraints on security architectures for MANETs. MANETs cannot always guarantee online access to a centralized CA due to the often intermittent and unreliable nature of the wireless channel. Thus, the use of a standard public key infrastructure (PKI) is generally infeasible in an ad hoc wireless environment.

The goal of securing ad hoc wireless networks has generated much interest in the research community in recent years. Password-based schemes for key establishment [1], [2], [3], [4], [5] avoid the need for a CA by carrying out authentication on the basis of a shared secret or password established prior to the deployment of the network. A security scheme similar to Pretty Good Privacy (PGP) has been proposed for MANETs [6], [7], whereby certificates are issued by users based on the establishment of chains of trust. This approach is well-suited to the ad hoc networking environment, but provides only probabilistic security guarantees and relies on transitive trust relationships, which may not be sufficient for some applications. Another approach to securing MANETs is based on a distributed certification authority (DCA) [8], [9] to avoid the problem of a single point of failure found in traditional PKI architectures. This approach provides deterministic security guarantees, but raises critical issues of scalability in practical MANETs. The concept of a distributed certification authority has also been applied to wired networks in a security architecture called COCA [10], which also provides fault tolerance. However, COCA cannot be directly applied to ad hoc networking environments where the behavior of the nodes is complicated by dynamic changes in wireless connectivity. In general, PKI architectures designed with wired networks in mind cannot be carried over straightforwardly to ad hoc networks [8].

We propose a comprehensive approach to providing a distributed CA-based PKI in MANETs, which potentially allows the network size to scale to hundreds or even thousands of nodes. The key elements of our approach are: (1) a scheme for dynamically partitioning the network into smaller clusters of nodes based on nodal mobility; (2) a distributed certification authority with multiple CA servers employing threshold-based cryptography with proactive share recovery, and replicated key repositories assigned to each cluster; (3) the use of elliptic curve cryptography (ECC). Distributing CA servers geographically over the coverage area of the MANET makes it more difficult for an adversary to compromise multiple CA servers simultaneously. Further, the use of threshold-based cryptography with proactive share recovery forces an adversary to simultaneously compromise more than half of the CA servers before the distributed CA itself is compromised. The distribution of the key management architecture over clusters reduces the storage requirements of the nodes and the CA servers, as well as the computational and signaling overhead. Finally, the use of elliptic curve cryptography dramatically reduces the computations involved in cryptographic operations, making the PKI-based scheme feasible even for nodes of modest computational power.

The main contributions of the paper are a practical architecture for implementing a distributed CA over a MANET via dynamic clustering and a detailed study of the computational gains achievable by using elliptic curve cryptography in the MANET setting. The proposed architecture provides a highly secure PKI with a flat trust management architecture and deterministic security guarantees. We discuss the advantages of the proposed architecture compared to existing schemes and how the architecture can resist a wide range of security attacks.

The remainder of the paper is organized as follows. Section 2 discusses the elements of the proposed distributed CA-based PKI architecture. Section 3 focuses on the operational aspects of the cluster-based key management protocols within the proposed PKI architecture. Section 4 provides a detailed analysis of the performance

gains achieved by using ECC in the MANET setting. Finally, the paper is concluded in Section 5.

2 Distributed CA-based PKI Architecture

2.1 Overview

Figure 1 gives a three-tiered logical view of how the DCA architecture is organized. At the lowest tier individual nodes are organized into clusters using standard clustering schemes [11], [12], [13], [14]. The next tier consists of one or more certificate repositories in each cluster. The top tier consists of DCA servers. Although the servers and repositories are represented at higher levels from the other nodes, the proposed scheme is in fact a flat PKI trust hierarchy. We next discuss the details of how these logical tiers are organized.

The number of DCA servers should be a function of the network size and the degree of resilience required against attacks. We utilize a threshold based scheme to govern this. The number of servers is defined by $n = 2k+1$, where k is the maximum number of servers that can be compromised in a predefined period of time. Each server participates in issuing certificates and revocation certificates (counter-certificates) and in periodically signing the certificate revocation list (CRL) that contains the serial numbers of revoked certificates from the entire network. The servers are assumed to be physically more secure and computationally more powerful nodes. The initial distribution of the CA servers is described in section 3.1. If a server is compromised but undetected, it is because it functions properly; in this case no measures are necessary. Once a server is compromised and detected, it cannot perform service as part of the DCA, until its share is recovered or renewed. It does not influence or affect the functioning of the cluster or the system.

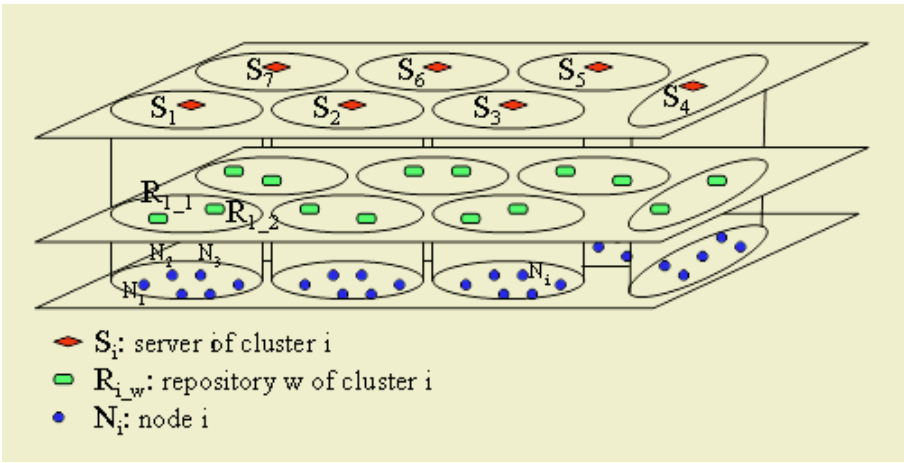


Fig. 1. DCA architecture ($n=7$, $t-r=2$)

Table 1. Certificates available to current and future participants of the system

Available to:	Cluster Server	Other Server	Cluster Nodes	Non-Cluster Nodes	New Nodes
Certificate	X		Optional		Optional
Counter-Certificate	X	X	X	X	X
Most recent CRL	X	X	X	X	X

Within each cluster, a fixed number t of nodes are designated as repositories that store the certificates of the nodes within the cluster, the certificates of all servers, the counter-certificates of the network nodes, and the most recent version of the CRL. The repositories might also become compromised and thus become unavailable (note that in case of compromised repositories nothing that was not already public is revealed). However, up to r repositories may be compromised within a cluster before a new node within a cluster is elected to serve as a repository. This way, there will always be a minimum of $t - r$ active repositories in each cluster.

The use of clustering has two advantages. First, it reduces the storage requirements of individual nodes as each node needs to store at most the certificates of the other nodes in the same cluster rather than the entire network. Second, clustering reduces the communication overhead and increases the efficiency of certificate management as certificates are always available to each node at a local repository within a small number of hops.

A key feature is that the CRL maintained in the repositories is timestamped, signed by the DCA, and updated every day. The corruption of the repositories is acceptable, since (1) corrupted certificates or counter-certificates will be detected via signature verification, (2) corruption that involves deletion of certificates and/or counter-certificates, is also detected, as the most recent CRL can be checked. If the information provided is up-to-date, it is considered correct. If it is not, another repository is accessed. Hence, the existence of $t - r$ active repositories in each cluster ensures that the cluster's operation is never interrupted.

2.2 Threat Model and Resistance to Attacks

Our focus is on the compromise of DCA servers and certificate repositories and the effects they will have on the integrity and availability of certificate management services. We consider the following attack model to characterize the resiliency of our schemes. We denote an attack type by a triple (c, s, r) , where c is the number of clusters involved, and s and r represent the number of DCA servers and repository copies, respectively, that can be compromised by an attack.

The first attack we consider is the well-known theft of the secret key of the CA. This represents a single point of failure in a CA infrastructure and allows an attacker to forge certificates. Our distributed implementation of the CA uses a threshold cryptography scheme to protect the secret key of the CA by distributing the secret

(private key) among a number of servers (shares) and thus avoids the single point of failure vulnerability. This scheme can thus be used to deal with attacks of the form (n, k, r) where k is the maximum number of servers compromised out of a total number of n servers with $n = 2k+1$, and there is at most one DCA server per cluster. It can also deal with attacks of the form (p, k, r) where we have a total of n servers with $n = 2k+1$ and $p < n$. In this case, some clusters may have more than one server.

We next consider attacks that reduce the availability of certificate repositories. Our cluster-based distributed CA scheme can handle attacks ranging from $(1, 0, r)$ to $(1, s, r)$ by maintaining at least t replicas (where $t > r$) of the certificate repository within each cluster. The number $b = t - r$, is tunable so as to always guarantee b copies of the repository. In the case of $(1, s, r)$, all s servers in a cluster are compromised and unavailable but as long as one replica of the repository is functioning we can provide all the necessary certificate services including the addition of a new member and the distribution of its certificate.

Another source of vulnerability in a CA system has to do with the authentication and validation of requests to issue certificates. In our scheme, this function is performed by a Registration Authority (RA). A common vulnerability is that the RA can be fooled into believing B when B impersonates A . We assume that the RA (1) does not belong to the MANET but is part of a wired network; (2) can communicate with the servers of the DCA securely; (3) does not know the private key of the DCA. The RA will verify the credentials of a node and if satisfied, contact at least $k+1$ servers and request issuance of a certificate. This model reflects the practical procedures and work scenarios present in many environments that use wireless networking. For example, before troops go out to the battlefield with their wireless devices, they will be required to report to and register at the RA. Upon successful verification of credentials, a wireless device with a private key and an associated certificate with the public key may be issued to a soldier to enable further communications in the battlefield.

Passive attacks such as eavesdropping of wireless communications are not a concern for us as any information gathered from such activities is public knowledge. Finally, the integrity of certificates through active tampering can be verified through the use of digital signatures.

2.3 Elliptic Curve-Based Distributed Certification Authority

We propose a distributed certification authority based on threshold cryptography and proactive secret sharing [15], [16]. The traditional public key cryptosystem employed in [15], [16] is impractical for MANETs, as it imposes high computational and communication overhead. Therefore, we propose the use of ECC [17] to reduce this overhead for the mobile devices. The DCA consists of $n = 2k+1$ servers that will share a secret value x , through a $(k+1, n)$ threshold scheme [18]. Any $(k+1)$ servers are required to combine their shares in order to sign a message, while the adversary who wants to learn or destroy its secret signature key has to compromise more than k servers during a single time period. A broadcast communication channel is assumed. The goal is to prevent the adversary from learning or destroying the secret x .

Time Periods and Update Phases. We assume that time is divided into time periods, during which the servers can perform the group signature operation. Each period is determined by a common global clock and its duration is specified as required (five days, two weeks, etc.). There are short update phases for the re-randomization of the original key. Previous shares become useless and should be erased, since combined information about two periods of the system could enable the adversary to break the system.

Each update phase consists of: 1) private key renewal, which renews the means of encryption and authentication among the servers; 2) lost share detection and recovery, which checks the current shares of the servers and reconstructs the corrupted shares, if any; and 3) share renewal, which re-randomizes the secret key x . The resolving accusation protocol [15] is implemented when two servers' claims are contradictory at specific phases of the protocols. This protocol might be completed in 3 or 4 steps, depending on the state of the transactions.

Cryptographic Tools. We assume that the signature scheme used by the DCA is Elliptic Curve El Gamal signatures, which is discussed in more detail in section 4. We assume that Feldman's verifiable secret sharing (VSS) scheme [19] is used for the sharing of the CA's key among the n participating servers. Each server has two pairs of keys, one for encryption and one for signature that will provide both private and authenticated communication. Table 2 shows the keys and parameters of the CA that are known to different elements of the system.

3 Operational Aspects of the Proposed Architecture

The notation used in this section is shown in Table 3.

3.1 Network Initialization

The proactive secret sharing system that we adopt for our DCA assumes an initialization phase, during which (1) the secret key of the CA is generated and shared between the servers, (2) each server generates its pair of authentication and encryption keys and publishes their public values to the group of servers, (3) an initial set of nodes is deployed. For simplicity we shall assume that the nodes form clusters, with one CA server per cluster. After the initialization phase, nodes may migrate between clusters and new nodes may join or leave the network.

3.2 Activating a Node

In the proposed scheme when a new N node wants to join the mobile network, it must carry out the following steps.

Table 2. Parameters of the DCA available to current and new participants of the system

Available to:	Server i	All Servers	Current Nodes	New Nodes
System parameters	X	X	X	X
Public key of DCA Y	X	X	X	X
Partial verification keys	X	X		
Share x_i of the secret x	X			
Server own signature key	X			
Server signature verification key	X	X	X	X
Server encryption key	X	X	X	X
Server decryption key	X			

- A1. Node N obtains certificate c_N
- A1.1. N contacts RA
 - A1.2. RA verifies credentials of N and securely contacts $k+1$ CA servers
 - A1.3. $(k+1)$ CA servers issue c_N for N and send it to RA
 - A1.4. RA gives c_N and c_{DCA} to N
- A2. Node N joins cluster i
- A2.1. N sends c_N to R_{i_w}
 - A2.2. N requests C_i , CC, CRL from R_{i_w}
 - A2.3. R_{i_w} broadcasts c_N
 - A2.4. R_i store c_N
 - A2.5. j_i optionally store c_N

In step A1, the node that wants to enter the network contacts a fixed Registration Authority. The RA will verify the credentials and if satisfied, contact at least $k+1$ servers and request issuance of a certificate. Then, $k+1$ servers will perform the distributed signature operation protocol to issue a certificate for the new member. The issued certificate will be sent to the RA, the new member will obtain it and join the network. The new member can join any cluster of the network since its certificate's signature can be verified by any server or node (the public key of the DCA is stored by all network participants).

Table 3. Notation used in this section

DCA	set of CA servers	S_i	set of servers of K_i
K	number of clusters	C_i	set of certificates of nodes in K_i
R	number of repositories	R_i	set of repositories in K_i
RA	Registration Authority	c_{i_w}	certificate of j_{i_w}
CC	set of counter-certificates	cc_{i_w}	counter-certificate of j_{i_w}
c_{DCA}	certificate of DCA	S_{i_w}	server w of K_i
J	number of nodes	R_{i_w}	repository w in K_i
j_i	number of nodes in K_i	j_{i_w}	node w in K_i
K_i	cluster i	N	a new node

The introduction of a newly certified node in a cluster is described in step A2. The node has to contact one of the repositories of the cluster it wishes to join in order to publicize its certificate, which will be broadcast to the current nodes and repositories of the cluster. The node can obtain the certificate of any node in the cluster by contacting a local repository. Usually the certificates that are most likely to be used should be cached. The local storage of certificates makes communication within a cluster efficient, even when encryption is needed. Moreover, the repository will provide the new node with the most recent version of the certificate revocation list (CRL) containing serial numbers of revoked certificates from the entire network signed by the DCA and the counter-certificates issued since the last update of the CRL.

3.3 Deactivating a Node

The certificate revocation process takes place as described by the following protocol.

- D1. if m nodes of K_i want cc_{i_w} to be issued
 - D1.1. m nodes of K_i send signed accusations about j_{i_w} to at least $k+1$ CA servers
 - D1.2. $k+1$ CA servers issue cc_{i_w} and add the serial number of cc_{i_w} to the CRL
 - D1.3. cc_{i_w} is broadcast to the network
 - D1.4. $S_{i_w}, R_{i_w}, j_{i_w}$ store cc_{i_w}
 - D1.5. CRL is renewed and timestamped by the DCA periodically
 - D1.6. $S_{i_w}, R_{i_w}, j_{i_w}$ store CRL
- D2. if node j_{i_w} wants to request revocation of its own certificate
 - D2.1. j_{i_w} sends a signed request for the issuing of cc_{i_w} to at least $k+1$ CA servers
 - D2.2. follow steps D1.2 to D1.6

Revoking a certificate for a given node can be initiated either by m users belonging to the same cluster, where m can vary depending on the application of the network, or by a node that wants to revoke its own certificate. When the revocation process is initiated by m users requesting revocation of node's j_{i_w} certificate, their request needs to be sent to at least $k+1$ servers. As a result, the DCA issues a counter-certificate and adds a serial number of the revoked certificate to the global CRL. The revocation certificates are broadcast to all nodes of the network, immediately after being issued. The storage of the counter-certificates is obligatory. All the servers and repositories keep the CRL that contains serial numbers of revoked certificates from the entire network. CRLs are renewed by the DCA every day or more often if necessary and are broadcast to all nodes in the network.

The revocation process may also be initiated by a node that wishes to revoke its own certificate either because it wants to leave the mobile network, or because its private key has been compromised. In this case, the node sends a signed request to at least $k+1$ servers to enable the issuing of its revocation certificate.

3.4 Node Migrations across Clusters

A highly mobile node j_{i_w} might leave source cluster K_s and enter destination cluster K_d . The following protocol describes the protocol to manage smooth node migrations across clusters.

- M1. Node j_{i_w} leaves source cluster K_s
 - M1.1 if the mobility management protocol indicates that the node j_{i_w} permanently leaves the source cluster K_s , j_{i_w} deletes the certificates C_s of the nodes in K_s
 - M1.2 else, go to step M2.
- M2. Node j_{i_w} joins destination cluster K_d
 - M2.1. j_{i_w} sends its certificate c_{i_w} to repository w of K_d (R_{d_w})
 - M2.2. j_{i_w} requests the certificates C_d of the nodes in cluster K_d from R_{d_w}
 - M2.3. R_{d_w} broadcasts c_{i_w}
 - M2.4. the repositories of K_d (R_d) store c_{i_w}
 - M2.5. the nodes of K_d (j_d) optionally store c_{i_w}

When node j_{i_w} leaves source cluster K_s and enters destination cluster K_d , it does not know the certificates of the nodes in cluster K_d . Therefore, it contacts any of the cluster K_d repositories (R_{d_w}) in order to obtain them. At the same time, node j_{i_w} is introduced to the cluster by sending its certificate to R_{d_w} . Besides that, node j_{i_w} can send its certificate to each node it corresponds with and the certificate can be authenticated using the public key of the DCA that each node in the network knows. The certificates of the nodes of the source cluster K_s that are stored in node j_{i_w} are deleted, unless the mobility management protocol predicts that the node is temporarily moved to a new cluster. In this case, the node can be programmed to delete the certificates of cluster K_s when it has moved to a third cluster K_h . Another option is to keep the stored certificates if enough storage space is available.

3.5 Intra-cluster Communications

Communication inside a cluster is relatively fast, regardless of whether the communication is encrypted or authenticated. This is because each node caches (1) the most frequently used certificates of the nodes within the cluster, (2) the revoked certificates from the entire network and (3) the most recent version of the CRL. Consequently, the nodes infrequently request certificates or counter-certificates from the repositories, hence reducing the communication overhead. The cluster's CA server periodically informs the cluster about the new network counter-certificates when they are issued and the updated CRL. Repositories broadcast the certificates of new nodes.

3.6 Inter-cluster Communications

The way inter-cluster communication takes place depends on whether it needs to be authenticated or encrypted. Since the public key of the DCA is known to all the

participants of the system, the certificate of any node can be verified by any other node. Thus, the authentication has very low communication overhead. On the contrary, when an encrypted message needs to be sent, the sending node does not know the public key of the receiving node, because it has only cached certificates of the same cluster. Then, the required certificate has to be requested from one of the repositories of the cluster to which the receiving node belongs. The knowledge of the counter-certificates of the whole network and the most recent CRL is an advantage, since all the nodes are aware of all the revoked certificates and as a result a revoked certificate will never be requested. This reduces the communication overhead. The reply that is sent contains the requested certificate. However, it should be noted that node, whose certificate is requested, might reply to the request route and might send the certificate by itself.

3.7 Cluster Splitting and Merging

The network size and distribution of nodes may change dynamically, which affects the number of clusters as defined by the cluster management protocol [11], [12], [13], [14]. For the purposes of the proposed DCA-based PKI architecture, we shall assume the basic functions of a generic cluster management protocol, i.e., cluster splitting and merging operations.

If the network size increases and new clusters are formed, the number of network clusters may become larger than the number of DCA servers, since the number of DCA servers is fixed $n = 2k+1$. In this case, not all clusters will contain one CA server. Assume, for example, that cluster i splits into two clusters i_1 and i_2 . The CA server of cluster i will become part of a cluster of higher density, for instance i_1 . Hence, cluster i_2 will not contain a CA server, but its nodes will be mapped to the server of cluster i_1 . The key issue is that the CA servers are distributed into the network. However, the cluster i_2 will need to elect t repositories. Any of the cluster nodes that have never been accused of misbehavior can serve as a repository.

If the network size decreases and some clusters are merged, the number of network clusters may become smaller than the number of DCA servers. In this case, a cluster that results from the merger of two clusters will contain the CA servers and repositories of the original clusters.

4 Performance Gains Using ECC

Having presented the distributed CA-based PKI scheme, we now discuss the performance improvement of the scheme through the use Elliptic Curve cryptography. ECC is appropriate for mobile nodes with limited computational power because it requires smaller keys and involves operations on smaller integers than in standard systems.

4.1 Cryptographic Tools

A sufficiently large prime p , an Elliptic Curve E over $GF(p)$ with a total number of points N and a generator of the group of points on the elliptic curve P are chosen. Let c be a certificate to be signed. The above settings are publicized. The private key can be any number x , where $1 \leq x \leq \#E(GF(p)) - 1$. Here, $\#E(GF(p))$ denotes the number of points on the curve. Its corresponding public key is a point $Y = x * P$.

The signature scheme used by the DCA is Elliptic Curve El Gamal signatures. For the elliptic curve-based system we choose to implement the Elliptic Curve Digital Signature Algorithm (ECDSA) signature scheme for signatures since it is a standard and the Elliptic Curve El Gamal encryption scheme for encryption. We have chosen a key size of 160 bits (i.e., p, q are 160 bits long for both the EC El Gamal and ECDSA), which is equivalent to a 1024 bit key in traditional public-key cryptosystems. A 160 bit key size provides a sufficient level of security for most applications, while placing a reasonable computational burden on the nodes of a MANET.

4.2 Computational and Time Requirements of the DCA

A detailed analysis of the computational and communication overhead for each protocol of the elliptic curve-based DCA scheme and the traditional public key system employed in [15], [16] is given in [20]. The overhead depends on the values of the variables k, β, m , where $k = (n-1)/2$ and n the number of servers, β the number of servers of faulty shares that are recovered (during the recovery of lost shares protocol) and m the number of servers that misbehaved during the execution of the share renewal protocol. The formulas derived in [20] show that the number of computations (of modular multiplications) to be performed and the traffic generated is essentially proportional to k^2 .

To illustrate the practicality of the elliptic curve-based DCA scheme on a large network, we shall consider a network of 51 servers ($k = 25$). We choose β and m to be equal to 2. We shall assume that each cluster contains approximately 100 nodes. As a result, the network contains about 5100 nodes and 51 servers.

To evaluate the elliptic curve-based system, it is crucial to compare it with the original, traditional public key system. Let EC-DCA denote our elliptic curve-based DCA scheme and T-DCA denote the original, traditional public key system. For the EC-DCA system, the results were derived as the number of modular multiplications with a modulus of 160 bits. For T-DCA, the results were derived in terms of the number of modular multiplications with a modulus 1024 bits. However, the time required to perform a modular multiplication with modulus of the size of z bits in software is proportional to the z^2 . Thus, modular multiplications with modulus of the size of 160 bits can be normalized to modular multiplications for a 1024 bit modulus, to simplify the comparison.

Table 4 presents the comparison of the computations required by each server for each protocol of the scheme when $n = 51$. Here, the unit of computation is one modular multiplication, where the modulus is determined by the key size. The various types of computation involved in the cryptosystem are presented in terms of the

number of equivalent modular multiplications required. We note that each server does not perform the same number of computations. Since not all of the servers contribute to the distributed signature operation, not all the servers need to recover their shares. Also, some servers may not participate if they are accused of cheating. Thus, we shall consider the maximum number of computations that have to be performed per server. Even though not all the servers will need to perform the maximum number of computations, the time needed to finish running a given protocol is determined by the servers that perform the most computations.

The number of computations that each device has to perform can be translated into the time that the device needs to perform those computations, taking into account the computational power of the present mobile devices. At present, smart cards can perform up to 3000 modular multiplications per second with the size of the modulus being 1024 [21]. Based on that, we can calculate how much time a server needs to perform the computations needed for each protocol. Table 4 presents a comparison of the maximum number of computations per server and the time required for the following protocols: (1) Distributed Signature Operation, (2) Private key renewal, (3) Lost share detection, (4) Recovery of lost shares, (5) Share renewal, (6) Resolving accusations - in 3 steps -, (7) Resolving accusations - in 4 steps -.

Table 4. Comparison of the *max* computations per server and the time required for each protocol ($n = 51$)

Max computations (modular multiplications) <i>per</i> server							
Protocol	1	2	3	4	5	6	7
EC-DCA scheme	13,314	8,574	17,578	4,694	16,915	164	2,221
T-DCA scheme	114,146	36,240	72,480	88,563	349,777	3,072	39,961
Time (in seconds)							
Protocol	1	2	3	4	5	6	7
EC-DCA scheme	4.44	2.86	5.86	1.56	5.63	0.05	0.74
T-DCA scheme	38.04	12.08	24.52	29.52	116.59	1.02	13.32

4.3 Communication and Storage Requirements

The communication overhead imposed by the proposed scheme is shown in Table 5. As the majority of involve messages that are broadcast, the number of messages sent differs from the number of messages received and thus processed within the DCA. The transmission requirements are manageable for 51 servers. The values of the variables involved are discussed in Section 4.2.

The key space is partitioned with the use of clusters and the use of small key sizes, since the system is elliptic curve-based. We calculate the key storage requirements for a network of 51 servers ($k=25$), 51 clusters and approximately 5100 nodes (about $j=100$ nodes per cluster). In particular, we compute the key space required for the storage of certificates, counter-certificates, CRL and DCA parameters per server,

repository and node. Nodes choose to store as many certificates of the same cluster's nodes as needed, whereas the repositories have to store the certificates of all nodes in the cluster. Therefore, the key space required per node varies. In case all 100 certificates of the cluster are stored, we can calculate the maximum key space required per node, which equals the key space required per repository. We also assume that no more than 20 counter-certificates are issued per cluster; thus 51*20 in all clusters. The size of the CRL, under the assumption that one serial number is 32 bits, is 4.08KB. It is computed that the maximum key space required for the storage of certificates, counter-certificates, CRL and DCA parameters per server, repository and node are 29.66KB, 28.6KB, and 28.6KB, respectively. It is apparent that the key space required is very small. This is one of the major advantages of our system.

Table 5. Traffic generated within the DCA for each protocol (n = 51)

Protocol	Messages sent	Messages received
Distributed Signature	78	3901
Private key renewal	51	2550
Lost share detection	102	5100
Recovery of lost shares	147	4900
Share renewal	102	5100
Resolving accusation - 3 steps -	2	100
Resolving accusation - 4 steps -	2	100

5 Conclusions

The proposed distributed CA-based PKI architecture addresses several key challenges in securing MANETs: (1) The physical vulnerability of the nodes in a hostile environment is addressed by employing the distribution of the CA's functionality across multiple nodes and using threshold cryptography with proactive recovery; (2) the insecurity of the wireless links is dealt with the use of keys so that the information exchanged is authenticated and encrypted; (3) the storage constraints are addressed with the use of ECC (the key size is reduced) and the use of clusters (the number of keys stored is reduced); (4) the energy constraints are addressed with the use of an ECC-based cryptosystem and clustering to reduce communication overhead. Finally, the use of clustering allows the proposed PKI scheme to scale to large networks. The proposed architecture could be implemented using current smartcard technology [21].

Acknowledgment

This work was supported in part by the U.S. National Science Foundation under Grant CCR-0209049.

References

1. N. Asokan, P. Ginzboorg, "Key Agreement in Ad-hoc Networks", Northsec 1999, Sweden.
2. S. M. Bellovin, M. Merrit, "Encrypted Key Exchange: Password-based protocols secure against dictionary attacks". In Proceedings of the IEEE Symposium on Research in Security and Privacy, 1992.
3. S. Lucks, "Open Key Exchange: How to defeat dictionary attacks without encrypting public Keys". In Security Protocol Workshop '97, Ecole Normale Suprieure, Paris, 1992.
4. D. P. Jablon, "Extended password key exchange protocols immune to dictionary attack". In Proceedings of the WETICE '97 Workshop on Enterprise Security, Cambridge, MA, USA, 1998.
5. T. Wu, "The secure remote password protocol". In Symposium on Network and Distributed Systems Security (NDSS '98), pages 97-111, San Diego, California, 1998. Internet Society.
6. J. P. Hubaux, L. Buttyan, S. Capkun. "The quest for security in mobile ad hoc networks". In Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2001.
7. S. Capkun, L. Buttyan, J.-P. Hubaux. "Self-Organized Public-Key Management for Mobile Ad Hoc Networks". Technical Report EPFL/IC/200234, Swiss Federal Institute of Technology, Lausanne, June 2002.
8. L. Zhou, Z. J. Haas, "Securing Ad Hoc Networks", IEEE Network Magazine, November 1999.
9. S. Yi, R. Kravets, "Key Management for Heterogeneous Ad Hoc Wireless Networks". Technical Report UIUCDCS-R-2002-2290/UIIU-ENG-2002-1734, University of Illinois at Urbana-Champaign, July 2002.
10. L. Zhou, F. Schneider, R. van Renesse, "COCA: A Secure Distributed On-line Certification Authority". Technical Report, Cornell University, 2000. Revised, 2002.
11. P. Basu, N. Khan, T.D. Little, "A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks", In Proceedings of Distributed Computing Systems Workshop, 2001.
12. S. Banerjee, S. Khuller, "A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks" In Proc. of IEEE INFOCOM, pages 1028-1037, 2001.
13. C.R. Lin, M. Gerla, "Adaptive Clustering for Mobile Wireless Networks", IEEE Journal of Selected Areas in Communications, vol. 15, no. 7, pages 1265-1275, 1997.
14. P. Krishna, N. Vaidya, M. Chatterjee, D. Pradhan "A cluster-based approach for routing in dynamic networks", Proc. of ACM SIGCOMM Computer Communication, pages 49-65, April 1997.
15. A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, "Proactive Secret Sharing or: How to Cope with Perpetual Leakage". In D. Coppersmith, editor, Proc. of CRYPTO'95, volume 963 of LNCS, pages 339-352, 1995.
16. S. Jarecki, "Proactive Secret Sharing and Public Key Cryptosystems", Master's Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, September 1995.
17. D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag, New York, 2004.
18. Y. Desmedt, Y. Frankel, "Threshold cryptosystems. Advances in Cryptology"- CRYPTO, LNCS 435:307-315, 1990.
19. P. Feldman, "A practical scheme for non-interactive verifiable secret sharing". In Proc. of IEEE Fund. Of Comp., Sci., pages 427-437, 1987.
20. C. Zouridaki, "Evaluation of the Proactive Public Key and Signature System and a new implementation based on Elliptic Curves", M.S. Thesis, Dept. of ECE, George Mason University, 2002.
21. W. Rankl and W. Effing, Smart Card Handbook, 2nd edition, John Wiley & Sons, Ltd., 2000.

ÆTHER: an Authorization Management Architecture for Ubiquitous Computing[±]

Patroklos G. Argyroudis and Donal O'Mahony

¹ Networks and Telecommunications Research Group, Department of Computer Science,
University of Dublin, Trinity College, Ireland
{argp, omahony}@cs.tcd.ie

Abstract. The ubiquitous computing paradigm suggests that we are going to be surrounded by countless wireless devices capable of providing services transparently. By definition, the nature of ubiquitous computing environments is open and extremely dynamic, making difficult the establishment of predefined security relationships between all of the participating entities. Authentication mechanisms can be employed to establish the identity of a pervasive computing entity but they suffer from scalability problems and have limited value in defining authorization decisions among strangers. In this paper we propose *ÆTHER*, an authorization management architecture designed specifically to address trust establishment and access control in ubiquitous computing environments. Owners define attribute authority sets and access control policy entries that are embedded into their devices. Members of the attribute authority sets are trusted to issue credentials for the corresponding attributes that can then be used in order to gain access to protected resources. Our architecture supports dynamic membership in these sets facilitating distributed administration, which is required in the context of the volatile nature of ubiquitous security relationships, and attribute mapping to allow roaming among authority domains. Moreover, we present the foundation of a logic model for our proposed architecture that is used to prove access control decisions.

1 Introduction

The technological developments of the last decade, particularly in the field of mobile microprocessor design, have enabled the integration of information processing capabilities in small everyday devices and appliances. This trend has led the computing world to a paradigm shift; the computer is no longer a personal dedicated device used for specific operations, but it is *woven* into the surrounding environment providing its services in a transparent manner. The ubiquitous computing paradigm envisioned by Mark Weiser [14] approaches computing from a human-centric, non-intrusive viewpoint: “The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.” The same vision is shared among many academic and industrial efforts that aim to realize

[±] This work is sponsored by the Irish Research Council for Science, Engineering and Technology (IRCSET), under contract number RS/2002/599-2.

S.K. Katsikas et al. (Eds.): EuroPKI 2004, LNCS 3093, pp. 246-259, 2004.

© Springer-Verlag Berlin Heidelberg 2004

it [7, 13]. However, the extremely dynamic and open nature of ubiquitous computing environments raises security and privacy concerns that need to be addressed in a coherent manner along with the development of the required infrastructure. Although the traditional security requirements remain the same, this new approach to computing has introduced additional challenges. The main problem in addressing the security requirements of pervasive environments is the large number of ad hoc interactions among previously unknown entities, hindering the reliance on predefined trust relationships. We envisage a (not too distant) future in which a user carries a multitude of devices that establish impromptu secure communication channels with the devices embedded into the environment or carried by other users. The problem is aggravated since data transmissions use wireless media, such as Bluetooth and IEEE 802.11b, whose integrity and confidentiality can easily be undermined by malicious entities. Furthermore, the proposed security solutions must take into account the ubiquitous computing vision that demands minimum user intervention for administrative tasks in order to be as non-intrusive as possible.

Traditional identity-based security infrastructures like X.509 can be used to establish the identity of a pervasive computing entity and provide authentication services by employing a centralized architecture. However, ubiquitous computing environments require decentralized administrating approaches in order to facilitate interactions with strangers. Another limitation of this approach lies in the use of identities as a basis for building trust and authorizing service requests. Identity-based credentials bind the owner's public key to a name. As we move to ubiquitous computing authority domains, naming becomes increasingly locally scoped. Therefore, a name may not have meaning to a different domain than the one that it was certified in. Scalability problems also exist when authorization decisions are based on identities in dynamic environments since enumeration of all the participating entities is nearly impossible.

Our proposed authorization architecture, named *ÆTHER*¹, is designed specifically to address dynamic ubiquitous computing environments where *a priori* knowledge of the complete set of participating entities and global centralized trust registers cannot be assumed. Users embed policy entries into their devices that allow access to protected services only to entities that present the correct attribute certificates (ACs), and definitions of attribute authority sets (AASs) whose members are trusted to certify the corresponding attributes. Our design supports dynamic membership in the AASs facilitating distributed administration and attribute mapping to allow roaming among authority domains. The idea of embedding access control policies into pervasive computing devices was originally proposed as part of the extended Resurrecting Duckling security model [11], however the authors did not provide any specific engineering details and only suggested the use of trust management systems.

In the remainder of this paper we start by briefly presenting the related work on this area in section 2. In section 3 we describe in detail our proposed architecture and define the foundation of the corresponding logic model. Section 4 discusses the strengths and shortcomings of our proposal before we conclude in section 5.

¹ The name was inspired by the medium that was once believed to pervade all space supporting the propagation of electromagnetic waves.

2 Related Work

Trust management, first defined in [6], addresses the problem of decentralized authorization by avoiding the use of identities and instead binds access rights directly to public keys. In such *key-based* access control systems, like PolicyMaker [6] and KeyNote [5], authorization certificates, implemented as signed assertions, are used to delegate permissions directly from the key of the issuer to the key of the subject. The syntax of the signed statements we use in *ÆTHER* is similar to the syntax of the signed assertions in KeyNote, but we bind attributes instead of access rights to keys in order to achieve flexibility and scalability in defining authorizations for a large number of entities.

A similar key-based effort that attempts to provide solutions to distributed authorization through the use of a public key infrastructure is SPKI/SDSI [8]. Its design is similar to that of KeyNote; the issuer of an SPKI/SDSI credential encodes the permissions it wishes to delegate to a subject using *s-expressions*, which are basically a set of constraints for issuer-defined values. In our proposed architecture we adopt the notion of identifying principals by their public keys instead of identities since it decouples the problem of authorization from the problem of secure naming. However, we argue that the delegation of a specific set of access permissions to a key is not appropriate for the ubiquitous computing paradigm in which the initiation of new services is frequent and the number of users particularly large. According to our view, a user that wishes to provide a new service should not be burdened with the additional administration overhead of initiating delegation chains and distributing the credentials that define the access rights for the new service.

An alternative approach for solving authorization problems more closely related to our own is the Trust Policy Language (TPL) [9]. An inference trust engine interprets TPL policies and the attribute credentials of previously unknown principals assigning them to access roles. Since the system allows the issuing of negative certificates that specify authorization restrictions it also provides a mechanism for collecting the credentials related to a principal. The main drawback of the TPL system is that it is designed for networking environments with a complete infrastructure and thus cannot be applied to the paradigm of ubiquitous computing. Furthermore, the use of negative certificates is particularly dangerous in open environments since if they are not collected in full the reached authorization decision might not be the expected one.

In the area of ubiquitous computing security the Resurrecting Duckling model was initially proposed to cover master-slave type of relationships between two devices [12]. The relationship is established when the master device exchanges over an out-of-band secure channel a secret piece of information with the slave device. Stajano and Anderson call this procedure *imprinting*. The imprinted device can authenticate the master through the common secret and accept service requests from it. The idea of bootstrapping trust relationships in adversarial environments over secure out-of-band channels was examined further by Balfanz *et al.* [4] and their concept of *location-limited channels*, which have the property that human operators can precisely control which devices are communicating with each other thus ensuring the authenticity requirement. They propose the use of contact and infrared as a way of exchanging pre-authentication information, such as the public keys of the peers, before the actual key

exchange protocol. The extended Resurrecting Duckling security model proposed the imprinting of devices with policies that define the type of relationships the slave device is allowed by its master to have with others in order to address peer-to-peer interactions [11]. However, the authors simply proposed the use of trust management systems as a way to define the imprinted policies without offering any specific implementation details. Furthermore, even the extended Resurrecting Duckling system defines a static association model between the master and the imprinted devices, limiting its direct application in situations where associations are established in an ad hoc manner. In ÆTHER we have extended these concepts and designed a complete authorization management system for dynamic pervasive computing environments.

3 Proposed Architecture

According to our vision the global ubiquitous computing infrastructure will be built in a non-hierarchical manner by connecting different autonomous domains via the existing mobile and fixed communication networks. The security considerations associated with this vision must be addressed in parallel with the development of the architecture designs that are going to support its employment. Consider the following simple but typical scenario in a pervasive computing environment (illustrated in Fig. 1). The owner wishes to control a light switch through her wristwatch via a wireless transmission medium and also wants to allow a trusted visitor in the house to do the same. However, in both cases she wants to be sure that a malicious party who walks outside the house will not be able to control the light switch and capture or modify any of the above wireless transmissions. Additionally, the necessary administrative actions should be kept to an absolute minimum.

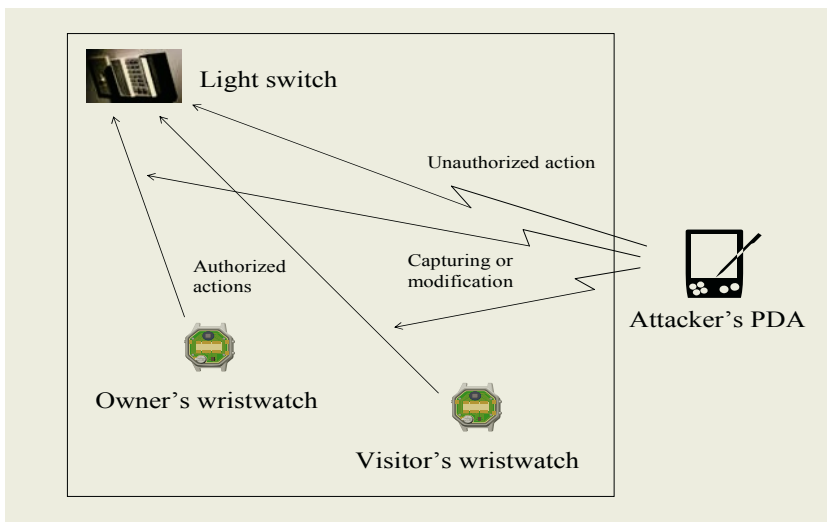


Fig. 1. Typical insecure ubiquitous computing scenario

Our proposed authorization management architecture provides a way for the owners of pervasive devices to specify autonomous authority domains and the security relationships that form the foundation of trust in them. This initial owner-specified trust is used to allow the creation of dynamic trust associations with new devices that come into the domain, without the need for tedious administrative tasks or the reconfiguration of previously bootstrapped devices. An *authority domain* (AD) in the terminology of *ÆTHER* is defined as the initial set of relationships between attributes and principals specified in a security policy. The owner of several pervasive devices creates an authority domain by specifying in a policy which principals are trusted to certify which attributes. The policy is embedded into the owner's devices via a location-limited channel such as an infrared link. Moreover, the owner creates policy entries for controlling what attribute credentials a principal must present in order to get specific access rights to a resource provided by a device. A principal in our design is the public key of a key pair that comes built-in with every device and uniquely identifies it. The specific policy constructs that define the attributes of the authority domain and the principals that act as sources of authority for these attributes are called *attribute authority sets* (AASs). In *ÆTHER* permissions are modeled as the rights of an AAS. We associate rights with actions, so possession of an authority attribute permits the certified principal to perform a certain action. The certification of the authority attributes is performed by the members of the corresponding AAS.

The AASs can be either static or dynamic. A static set can only have as sources of authority the principals specified in the initial policy entry. The set of principals that act as sources of authority for dynamic AASs can grow without requiring the explicit change of a policy entry or the issuing of a new one. Hence, we provide distributed administration of ADs and facilitate the effortless introduction of previously unknown principals. Furthermore, we provide a mechanism to allow the linking of ADs by mapping corresponding attribute sets for supporting secure interdomain interactions.

3.1 Attribute Credentials and Policy Statements

In this section we give a detailed description of the primary data structures of our architecture, namely attribute credentials and policy statements.

Definition 1. Attribute. An attribute is defined as an ordered 2-tuple of the form *(name, value)* where *name* is the identifier of the attribute and *value* is its value.

Example 1. An attribute certificate that binds the attribute (group, family_object) to the principal Key4, certified by the principal Key0 is shown below (Key4 and Key0 are both public keys, Key0 could be the key of the owner's wristwatch and Key4 the key of the light switch from the previously presented scenario, the public keys and the signature are omitted for readability reasons):

```
aether version: 1
type: attribute certificate
issuer: Key0
```

```
subject: Key4
attribute name: group
attribute value: family_object
not valid before: 2004/02/02-12:21
not valid after: 2004/02/02-12:31
signature: Key0's signature
```

When the principal Key0 wishes to issue an attribute certificate for the principal Key4 the two devices establish a location-limited channel, for example an infrared link, with the help of their human owners. The attribute credentials that a principal possesses are used to support access requests to devices that provide services. In our current design we do not support certificate revocation lists (CRLs) and instead we rely on short expiration time periods and refreshing mechanisms for the issued credentials. The typical validity period we suggest is 10 minutes. If the subject is within the communication range of the issuer the certificate can be refreshed over a wireless transmission automatically before it expires. Alternatively, the subject principal must again establish a location-limited channel with the issuer to get a new certificate.

In ÆTHER we have three different kinds of policy statements. The *binding* policy statement serves the purpose of binding the subject principal to the issuing principal. This is similar to the imprinting functionality of the Resurrecting Duckling system, but instead of sharing a common secret we utilize a signed statement to achieve the same goal. The embedding of the binding policy entry to the subject device must take place over a location-limited channel in order to ensure its authenticity. After a device has been bound, it can accept new policy entries remotely by the principal that has bound it. If an owner wishes to revoke the binding of a device we assume that some physical means are provided by the manufacturer. We are currently investigating less cumbersome revocation mechanisms for binding policies.

Example 2. A binding policy statement that binds the principal Key4 to principal Key0 is represented as:

```
aether version: 1
type: binding policy
issuer: Key0
subject: Key4
signature: Key0's signature
```

The second kind of policy statement in our system, called *access control policy* (ACP) statement, is the one that defines the resources that are provided by the device and the attribute credentials that are required to access them.

Definition 2. Action. An action is defined as an ordered 2-tuple of the form (*resource*, *operation*) where *resource* is the identifier of the provided service and *operation* is the identifier of the operation allowed on the protected service.

Example 3. Returning back to the example of the remotely controlled light switch, an ACP statement that specifies that the action (switch, change_state) can be performed only by principals certified to have the attribute (group, family_member) or (group, visitor) can be the one shown below:


```

aether version: 1
type: access control policy
issuer: Key0
resource: switch
operation: change_state
attribute: (group, family_member) || (group, visitor)
signature: Key0's signature

```

The third kind of policy statement supported by *ÆTHER* is the one that specifies which principals are trusted to certify which attribute credentials. These policy entries, as well as the closely related concept of authority domains, are analyzed in the following subsection.

3.2 Authority Domains and Attribute Authority Sets

An authority domain (AD) is a logical construct that represents an administrative domain of ubiquitous computing devices, and more formally the set of initial statements that specify which principals are the sources of authority for which attributes. Example ADs include a household, an office, a classroom and a restaurant. We assume that ADs are administered by the human owners of the devices that comprise them.

An attribute authority set (AAS) is a policy statement that identifies the principals that are the sources of authority for a specific attribute. Membership in an AAS means that the corresponding principal is trusted to issue attribute credentials of the specific type. A member of an AAS is defined as a principal that has been explicitly denoted in the signed AAS statement as such, or an unknown principal that has been given the required ACs by at least a threshold number of members of the authority set. This threshold number is called *membership threshold value* and the accepted values are integers greater or equal to 2. This is in essence a delegation of the authority specified in the policy statement. The allowed depth of delegation is controlled by a *delegation depth* entry in the AAS definition statement that in essence implements *integer delegation control*. Although there have been arguments in the literature against integer control regarding the inability to predict the proper depth of delegation [8], the application domain of *ÆTHER* makes its use particularly attractive. The owner of an authority domain can use representative values for integer delegation control when he defines the AASs of the domain according to the importance of the attributes they authorize. Important attributes that convey a high value of trust can have small delegation depth values while more general attributes that authorize less important actions can have bigger values.

Example 4. In a domestic pervasive computing scenario the AAS policy statement for the attribute (group, visitor) that authorizes visitors to the house could specify a delegation depth and a membership threshold value of 2. This would allow a trusted visitor to introduce another visitor to the house, and in order for a visitor to be able to do so she must be trusted by two existing members of the AAS. This AAS policy statement with principals Key0 and Key1 as the initial sources of authority can be represented as:


```
aether version: 1
type: attribute authority set
issuer: Key0
attribute name: group
attribute value: visitor
membership threshold value: 2
delegation depth: 2
sources of authority: Key0, Key1
signature: Key0's signature
```

A delegation depth of -1 means that delegation of the specified authority is not allowed, a value of 0 means that delegation is allowed with no restrictions on the depth and any integer greater or equal to 1 defines the allowed depth. An AAS that has a delegation depth of -1 and a membership threshold value of 0 is defined as *static*, meaning that the set of principals that act as sources of authority for the corresponding attribute is not allowed to grow dynamically.

3.3 Secure Interdomain Interactions

One of the goals of our architecture is to enable secure interactions between principals of different authority domains. While a user roams administrative domains the devices he carries must be able to interact with the devices of the environment in a secure manner without requiring reconfiguration. When ACs are exchanged between different domains, attribute mapping mechanisms are needed to allow attributes from foreign domains to be translated into corresponding attributes in the domain where the AC is validated [10].

Although a global attribute registry has been suggested as a possible solution to the problem of attribute mapping [10], we believe that the introduction of a centralized universally trusted principal is not applicable to the highly distributed nature of ubiquitous computing. In ÆTHER we use *attribute mapping certificates* (AMCs) to map AASs of different authority domains.

Example 5. Consider the case of a business agreement between restaurants R1 and R2 that specifies that the privileged customers of R1 are to be accepted as normal customers in R2. This means that we want to map the attribute (customer, privileged) of R1 to the attribute (customer, normal) of R2. If the sources of authority for (customer, privileged) in R1 are the principals A1 and A2 and for (customer, normal) in R2 the principals B1 and B2 then the AMC that performs the mapping would have to be issued by B1 or B2. This AMC issued by principal B1 is shown below:

```
aether version: 1
type: attribute mapping certificate
issuer: B1
attribute name: customer
attribute value: normal
subject attribute name: customer
subject attribute value: privileged
subject sources of authority: A1, A2
not valid before: 2004/02/17-18:09
not valid after: 2004/02/18-17:09
signature: B1's signature
```

Obviously this is a one-way mapping. If a two-way mapping is required then another AMC must be issued, by A1 or A2, to map the attribute (customer, normal) of R2 to the attribute (customer, privileged) of R1. Moreover, the mapping is between the sources of authority of the two AASs, meaning that ACs issued by principals that dynamically joined the mapped set are not accepted as valid in R2.

3.4 Naming and Service Discovery

Although the use of public keys as identifiers for pervasive computing devices provides a secure binding between the device and its identifier, it suffers from usability problems. It is especially difficult (if not impossible) for humans to memorize the public keys of their devices. Hence, names are required in order to allow users to specify names instead of keys in interactions with their devices. Our attribute authority sets can be used to provide a mechanism to bind names to public keys. An authority domain can have an AAS responsible for providing naming credentials, exactly like any other attribute. The only difference is that the value field of the attribute is not included in the AAS policy statement but is specified when the owner uses one of the member devices of the set to issue a naming credential. All the devices that are part of the domain and have the AAS policy statement that defines the naming attribute can validate naming certificates and perform the mapping between public keys and user-specified names.

Service discovery is also an essential element of a pervasive computing architecture. Users must be able to identify the services that are provided in a domain and select the ones that are appropriate to the purpose they are trying to achieve. However, if service discovery is designed without a protection mechanism a malicious entity can easily gather information about the services that are provided in a domain. The profiling of the available services is usually the first step in an intrusion attempt and moreover is a violation of the privacy requirement; hence we must be sure that only authorized principals are able to get detailed resource advertisements regarding the devices of a domain. In *ÆTHER* the principals that are offering services periodically broadcast *public resource advertisements* (PRAs). A PRA is statement signed by the principal that transmits it which simply states that a service is provided. The idea is to treat service discovery as any other service that a device offers and must be protected. The principals that provide services have access control policy statements that specify what attributes are required in order to provide to a requesting principal a *detailed resource advertisement* (DRA). DRAs are also signed statements that present the precise actions that a device supports. Thus, a requesting principal that looks for a specific resource receives PRAs and based on the name of the advertising principal decides whether to inquire for more details. If the requesting principal provides the necessary ACs a DRA is disclosed and based on it a specific service request can then be assembled.

3.5 Logic Model

The authorization process in ÆTHER works in a similar way as existing trust management systems. A principal that makes an access request provides all the relevant attribute credentials it possesses along with the signed statement that describes the request. The validating principal verifies the presented credentials and passes them along with the signed policy statements it already possesses and the request to an inference engine that outputs whether the access request is allowed or denied. We have defined the foundation of a logic model for our architecture that can be used to prove an access control request based on the set of available signed statements. Our model is based on similar previous efforts to formalize trust establishment [1, 3], but focuses on the use of authorization attributes and attribute authority sets.

The expressions of our logic model are signed statements that are represented in the form of *Principal : Statement*, where *Principal* is a public key that signs the statement *Statement*.

Definition 3. Statement. A statement can be one of the following:

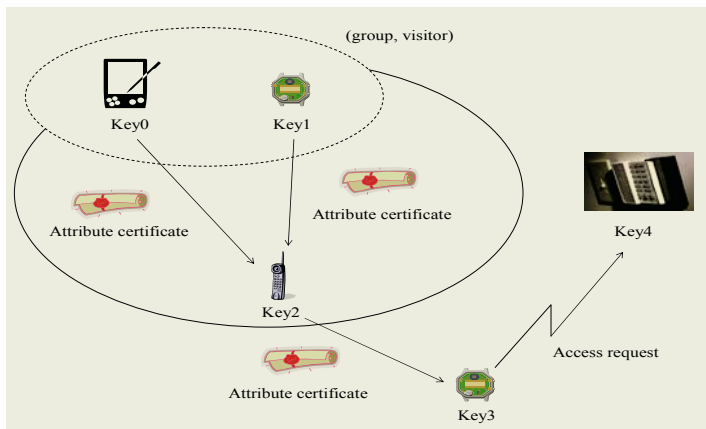
- *HasAttribute(K, A)*. This is a representation of the attribute credentials. It means that the principal that makes the statement certifies that principal K has attribute A.
- *Bind(K)*. This is a binding policy statement that represents the binding between the principal that makes the statement and the bound principal K.
- *Allow(R) → A*. This represents an access control policy. The principal that makes the statement allows action R if the requesting principal has attribute A.
- *AttributeAuthoritySet(A, S, D, T)*. An attribute authority set statement. A is an attribute, S is the set of principals that are the sources of authority for this attribute, D is the delegation depth and T is the membership threshold value.
- *AttributeAuthoritySetMember(A, K)*. This statement evaluates to the principal K (a public key) and has the meaning that K is a member of the attribute authority set A.
- *Map(A, B)*. This is an attribute mapping certificate. ACs certified by AAS B are accepted as though they have been certified by a principal that is a source of authority of AAS A.
- *Allow(R)*. This statement has the special meaning that the principal that makes it allows action R.

Furthermore, our logic model has four inference rules that can be used to deduce new statements from statements already known to be true. When a device receives a service request it passes to its access control inference engine all the valid signed statements. Based on the set of statements and the rules, the engine tries to deduce if the principal that bound the device allows the request. This is represented by the statement $K : Allow(R)$, where K is the principal that bound the device that receives the access request described by action R. If the inference engine succeeds in reaching this statement then the requested action is allowed, otherwise it is denied. The inference rules of our logic model are shown in Table 1.

Table 1. The inference rules of the $\mathcal{A}\mathcal{E}\mathcal{T}\mathcal{H}\mathcal{E}\mathcal{R}$ logic model

Name	Definition
Static membership rule	$\begin{aligned} & \text{If } X : A \text{ and} \\ & A = \text{AttributeAuthoritySet}(V, S, D, T) \text{ and } K \in S \\ & \text{then} \\ & K = \text{AttributeAuthoritySetMember}(A, K) \end{aligned}$
Dynamic membership rule	$\begin{aligned} & \text{If } X : A \text{ and} \\ & A = \text{AttributeAuthoritySet}(V, S, D, T) \text{ and} \\ & C = \{Y \mid \text{AttributeAuthoritySetMember}(A, Y) : \\ & \quad \text{HasAttribute}(K, V)\} \text{ and} \\ & C = T \text{ then} \\ & K = \text{AttributeAuthoritySetMember}(A, K) \end{aligned}$
Access rule	$\begin{aligned} & \text{If } X : R \text{ and} \\ & Y : \text{Allow}(R) \rightarrow V \text{ and} \\ & Y : A \text{ and} \\ & A = \text{AttributeAuthoritySet}(V, S, D, T) \text{ and} \\ & Z : \text{HasAttribute}(X, V) \text{ and} \\ & \text{AttributeAuthoritySetMember}(A, Z) \text{ then} \\ & Y : \text{Allow}(R) \end{aligned}$
Attribute mapping rule	$\begin{aligned} & \text{If } X : A \text{ and} \\ & A = \text{AttributeAuthoritySet}(V_A, S_A, D_A, T_A) \text{ and} \\ & Y : B \text{ and} \\ & B = \text{AttributeAuthoritySet}(V_B, S_B, D_B, T_B) \text{ and} \\ & X \in S_A \text{ and} \\ & \forall Z \in S_B : \text{AttributeAuthoritySetMember}(Z, A) \\ & \text{then} \\ & X : \text{Map}(A, B) \end{aligned}$

As an example application of the $\mathcal{A}\mathcal{E}\mathcal{T}\mathcal{H}\mathcal{E}\mathcal{R}$ logic model consider the following scenario (shown in Fig. 2), a simplified version of which has already been presented.

**Fig. 2.** Example authorization scenario

The principal Key0 has bound principal Key4, the remotely controlled light switch, and has issued an AAS policy statement for the attribute (group, visitor) and an access control policy:

$$\text{Key0} : \text{Bind}(\text{Key4}) \quad (1)$$

$$\text{Key0} : A \text{ where } A = \text{AttributeAuthoritySet}((\text{group}, \text{visitor}), \{\text{Key0}, \text{Key1}\}, 2, 2) \quad (2)$$

$$\text{Key0} : \text{Allow}((\text{switch}, \text{change_state})) \rightarrow (\text{group}, \text{visitor}) \quad (3)$$

The principal Key3 makes the following access request to the principal Key4:

$$\text{Key3} : (\text{switch}, \text{change_state}) \quad (4)$$

Moreover, the principal Key3 presents the following attribute credentials with its request:

$$\text{Key0} : \text{HasAttribute}(\text{Key2}, (\text{group}, \text{visitor})) \quad (5)$$

$$\text{Key1} : \text{HasAttribute}(\text{Key2}, (\text{group}, \text{visitor})) \quad (6)$$

$$\text{Key2} : \text{HasAttribute}(\text{Key3}, (\text{group}, \text{visitor})) \quad (7)$$

We also define C as the following set of attribute credentials of principal Key2 (actually the statements (5) and (6)):

$$C = \{ \text{Key0} : \text{HasAttribute}(\text{Key2}, (\text{group}, \text{visitor})), \text{Key1} : \text{HasAttribute}(\text{Key2}, (\text{group}, \text{visitor})) \} \quad (8)$$

Since Key4 was bound by principal Key0 the goal is to prove the statement $\text{Key0} : \text{Allow}((\text{switch}, \text{change_state}))$ by utilizing the above statements and the rules of our logic model. The proof is shown in Table 2.

4 Discussion

One of the main advantages of using authorization attributes instead of capability-based credentials is that certificate distribution and initiation of delegation chains is not required when a new device is bought. The owner simply embeds the required policy statements into the new device and the principals that already have the required attribute credentials can start using it. Moreover, the binding of principals to authorization attributes facilitates interactions with a large number of principals making ÆTHER scalable since there is no need for enumerating the relationships between all principals and actions.

Table 2. Proof of the authorization decision scenario illustrated in Fig. 2

Statement #	Statement	Proven by	From
(9)	$Key0 = AttributeAuthoritySetMember(A, Key0)$	Static membership rule	(2)
(10)	$Key1 = AttributeAuthoritySetMember(A, Key1)$	Static membership rule	(2)
(11)	$C = \{AttributeAuthoritySetMember(A, Key0) : HasAttribute(Key2, (group, visitor)), AttributeAuthoritySetMember(A, Key1) : HasAttribute(Key2, (group, visitor))\}$	Substitution	(8), (9), (10)
(12)	$ C = 2$	Cardinality	(11)
(13)	$Key2 = AttributeAuthoritySetMember(A, Key2)$	Dynamic membership rule	(2), (11), (12)
(14)	$Key0 : Allow((switch, change_state))$	Access rule	(2), (3), (4), (7), (13)

Although the direct use of public keys in the $\text{\textit{\textbf{ETHER}}}$ attribute credentials provides a level of privacy since the identity of the principal is not revealed, the structure of trust relationships in an authority domain can be disclosed by a malicious entity that observes the exchanges of attribute certificates. Further work is required on the privacy aspects of our architecture.

The design of the policies that describe an authority domain is a task that we would like to make simple for non-technically inclined users. Towards this goal we are developing default policy statements that can be used for the formation of authority domains in different pervasive computing situations. Manufacturers can provide these default sets of policies with their devices according to the services they provide and the context in which they are going to be used.

Another important issue for all credential-based systems is that of authority revocation. Our current design relies on short expiration time periods and refreshing mechanisms for the issued attribute credentials. Revocation is one of the areas that we plan to investigate further.

5 Conclusion

In this paper we have presented our ongoing work on the design of an authorization architecture that addresses the specific requirements of access control and trust establishment in pervasive computing. During our initial investigation of the problem domain we completed a thorough performance analysis of three of the most commonly used security protocols for networking applications, namely SSL, S/MIME and IPsec, on handheld devices [2]. The results of this work show that the time taken to perform cryptographic functions is small enough not to significantly impact real-time

mobile transactions in ad hoc environments. Therefore the overhead of the cryptographic processes used in ÆTHER is no obstacle to its implementation on modern handheld devices.

Although considerable work remains to be done both in the design and the implementation of our framework, we believe that ÆTHER provides a solid foundation upon which further research can be based for the development of a security architecture for pervasive computing.

References

1. Abadi, M., Burrows, M., Lampson, B., Plotkin, G.D.: A calculus for access control in distributed systems. *ACM Trans. Programming Languages and Systems*, 15(4):706-734, 1993.
2. Argyroudis, P.G., Verma, R., Tewari, H., O'Mahony, D.: Performance analysis of cryptographic protocols on handheld devices. Technical report TCD-CS-2003-46, University of Dublin, Trinity College, 2003.
3. Balfanz, D., Dean, D., Spreitzer, M.: A security infrastructure for distributed java applications. In *Proc. 2000 IEEE Symposium on Security and Privacy*, pp. 15-26, 2000.
4. Balfanz, D., Smetters, D.K., Stewart, P., Wong, H.C.: Talking to strangers: authentication in ad hoc wireless networks. In *Proc. 9th Network and Distributed System Security Symposium*, 2002.
5. Blaze, M., Feigenbaum, J., Keromytis, A.D.: The KeyNote trust management system version 2. Internet Engineering Task Force RFC 2704, 1999.
6. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In *Proc. 1996 IEEE Symposium on Security and Privacy*, pp. 164-173, 1996.
7. Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S.: EasyLiving: technologies for intelligent environments. In *Proc. 2nd Int'l. Symposium on Handheld and Ubiquitous Computing*, pp. 12-29, 2000.
8. Ellison, C., Frantz, B., Lampson, B., Rivest, R.L., Thomas, B., Ylonen, T.: SPKI certificate theory. Internet Engineering Task Force RFC 2693, 1999.
9. Herzberg, A., Mass, Y., Mihaeli, J., Naor, D., Ravid, Y.: Access control meets public key infrastructure, or: assigning roles to strangers. In *Proc. 2000 IEEE Symposium on Security and Privacy*, pp. 2-14, 2000.
10. Linn, J., Nystrom, M.: Attribute certification: an enabling technology for delegation and role-based controls in distributed environments. In *Proc. 4th ACM Workshop on Role-Based Access Control*, pp. 121-130, 1999.
11. Stajano, F.: The resurrecting duckling – what next? In *Proc. 8th Int'l. Workshop on Security Protocols*, LNCS 2133, pp. 204-214, 2001.
12. Stajano, F., Anderson, R.: The resurrecting duckling: security issues in ad hoc wireless networks. In *Proc. 7th Int'l. Workshop on Security Protocols*, LNCS 1796, pp. 172-182, 2000.
13. Want, R., Schilit, B.N., Adams, N.I., Gold, R., Petersen, K., Ellis, J.R., Goldberg, D., Weiser, M.: An overview of the PARCTAB ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28-33, 1995.
14. Weiser, M.: The computer for the twenty-first century. *Scientific American*, 265(3):94-104, 1991.

Trustworthy Accounting for Wireless LAN Sharing Communities

Elias C. Efstathiou and George C. Polyzos

Department of Computer Science
Athens University of Economics and Business
Athens 104 34, Greece
{efstath, polyzos}@aueb.gr

Abstract. The Peer-to-Peer Wireless Network Confederation (P2PWNC) is designed to be an open self-organizing Wireless LAN (WLAN) roaming association where users from one WLAN can access all other WLANs belonging to the same P2PWNC. Unlike other WLAN roaming schemes, P2PWNC is open to all types of WLANs and particularly to residential networks owned by individual householders. Without an identity certifying authority, trustworthy accounting of transactions in the P2PWNC is challenging, but accounting is necessary in order to enforce the basic P2PWNC ‘rule of reciprocity’. We show that even though the P2PWNC accounting mechanism and its purpose-built Public Key Infrastructure are open to Sybil attacks, there exists a user authentication algorithm that excludes all free riders and that can also make the percentage of unfair exclusions it causes very small simply by using more system memory.

Keywords. self-organized security, WLAN roaming, Sybil attack, peer-to-peer

1 Introduction

1.1 Wireless LAN Roaming Today

Wireless LAN (WLAN) technology based on the IEEE 802.11 family of standards is a success. Newer laptops and PDAs are commonly WLAN-enabled and 802.11 radios are also becoming part of consumer electronics devices such as digital cameras, MP3 players, and cellular phones [1]. Nevertheless, the economic value of these devices is greatly reduced because WLAN coverage is still not ubiquitous. This is unfortunate because it is relatively easy to set up a WLAN ‘hotspot’, i.e., an area where wireless Internet connectivity is provided: one only requires a WLAN access point / router and a broadband Internet connection, both of which are extremely small investments nowadays. However, the lack of universal roaming standards has resulted in a fragmented market of commercial public WLAN operators. Without roaming support, operators can provide only limited coverage, which makes it difficult to attract new customers and to fund additional investments in infrastructure. On the non-commercial side, the majority of residential, business, and university hotspots are closed to outsiders, and the various free hotspots available in several cities cover only a small number

of locations. Efforts are underway to establish WLAN roaming standards that would allow customers from one operator to access hotspots belonging to other operators. These efforts focus both on the technical [2] and on the roaming settlement issues [3]. In parallel, several *hotspot aggregators* [4, 5] are attempting to unite smaller operators by coordinating AAA and roaming settlement. These efforts still result in relatively ‘spotty’ coverage even for the largest of aggregators. Also, these solutions focus on commercial hotspot operators and are inappropriate, for example, for residential hotspots owned by individual householders.

1.2 A Peer-to-Peer Approach to Wireless LAN Roaming

In [6] we proposed a novel cooperative application called *Peer-to-Peer Wireless Network Confederation* (P2PWNC). P2PWNC is a system designed to unite independent hotspots in a WLAN sharing community where users that are associated with one hotspot can also access all other community hotspots. In P2PWNC, participants own and operate their local WLANs, but they also share them with roaming P2PWNC visitors that happen to be in range, much like a file-sharing community shares files. In principle, P2PWNC can incorporate various types of WLAN operators but its focus is on residential hotspots owned by individual households in an urban setting. With sufficient hotspots, built with independent and small investment decisions, we envisage citywide infrastructures through which P2PWNC participants would enjoy ubiquitous wireless Internet access. In [6], a number of issues involved with building a P2PWNC were presented and the peer-to-peer approach to roaming, where participants manage the infrastructure without relying on central authorities, was compared to existing roaming schemes and was presented as a simpler alternative to designs that attempt to replicate the complexity of cellular roaming.

The distinctive characteristics of P2PWNC are: (a) P2PWNC is completely self-organizing, requiring no central authorities, unlike the hotspot aggregation model which requires an aggregator; (b) P2PWNC is open to all hotspots that are willing to participate, unlike operator-only roaming associations that require legally binding roaming agreements; (c) joining P2PWNC is similar to joining a file-sharing community: there is no real-world communication or administrative overhead and only local actions (setting up a hotspot managed by a local software agent) are required; and (d) in the P2PWNC ‘market’ the only currency is payment in kind: as long as a participating hotspot is open to P2PWNC visitors, its owners can access other P2PWNC hotspots when roaming. We will call this last characteristic the *rule of reciprocity*.

In [7], P2PWNC was presented in economic terms as a *mechanism design* problem. There, the P2PWNC rules were formalized and, by using an analytic model, it was shown that as the number of P2PWNC participants becomes large, the optimal contribution policy may be approximated by a simple rule: any participant wishing to roam must be contributing a fixed amount of resources to visitors. This could be a fixed amount of Internet bandwidth in the local hotspot that is reserved for visitors. Enforcing this rule means participants that do not follow it must be *excluded* from consuming community resources.

1.3 Contributions of This Paper

In this paper we show that the P2PWNC reciprocity rule can be enforced and that free riders¹ can be excluded in a secure way even when all P2PWNC participants have equivalent roles and no authority controls system operation, not even during node initialization when new hotspots join P2PWNC. We assume that adversarial participants can tamper with P2PWNC components in an attempt to deviate from P2PWNC protocols. We assume that the main goal of an adversary is to attempt to benefit from the P2PWNC without contributing, i.e., by not operating a P2PWNC hotspot at all or by keeping the hotspot closed to visitors. We will also assume, however, that standard cryptographic assumptions hold, and that all private keys remain private.

We base our solution for trustworthy accounting of P2PWNC actions on a purpose-built Public Key Infrastructure (PKI) with no centralized Certification Authority (CA). We rely on signed receipts for accounting P2PWNC transactions, which are stored in a *decentralized storage subsystem*. We define a universal system parameter, the Time-to-Live (TTL), after which receipts expire and can be deleted from the system: the TTL is therefore a measure of the system's memory.

We claim that a P2PWNC would exhibit a *small world phenomenon* [9] and with the help of simulations we show how the fundamental *Sybil attack*² problem can be circumvented via a P2PWNC-specific authentication algorithm. We link the TTL to system performance, which we express as the number of *unfair exclusions* from consuming community resources. We show that the number of unfair exclusions can become very small and that there exists a TTL value above which there is no appreciable gain in system performance. Finally, we motivate future work on the P2PWNC by showing how our authentication algorithm can be made efficient.

2 System Model

2.1 Entities and Identities, Trust and Authentication

In P2PWNC, WLANs are referred to as *administrative domains*, or simply *domains*. Each domain has an associated set of *registered users* (e.g., for residential hotspots the registered users are the household members). The registered users, when accessing the Internet from their *home domain*, are considered local users and P2PWNC is not involved in their actions. When, however, *roaming users* access WLAN services from the *visited domain* within which they happen to roam, a P2PWNC *transaction* takes place. The visited P2PWNC domain is the *provider* and the home P2PWNC domain of the roaming user is the *consumer* in this transaction.

We define *altruistic domains* and say that a domain acts altruistically if it allows visiting users to access the WLAN without checking if their home domain follows the re-

¹ Free riding: Consuming community resources without contributing, a problem analyzed in the context of peer-to-peer systems in [8].

² Sybil attack [10]: A fundamental problem in open self-organizing systems with no identity-certifying authorities. An entity can appear in the system with multiple identities and invalidate any number of system assumptions.

ciprocity rule. We say that a domain is *unfairly excluded* when, although it does follow the reciprocity rule, one of its registered users is denied access by a visited domain. The reciprocity rule states that participants must make local resources available for consumption by other participants.

A P2PWNC *Domain Agent* (DA) represents each domain. P2PWNC is a peer-to-peer network of DAs that communicate over the Internet. DAs are software programs that can run on various types of physical hosts. For residential hotspots, we can assume that the DA runs on the wireless router itself and that it accesses the Internet (and other DAs) over a broadband connection. DAs automatically (a) regulate WLAN service provision to visitors by enforcing the reciprocity rule, and (b) form P2PWNC's decentralized storage subsystem. The DA network is completely self-organizing and the network is always open to new DAs (i.e., new WLANs).

We assume that each DA, upon initialization, generates a unique public-private key pair and the DA's public key serves as the domain's P2PWNC name. Users of domains also generate their own key pairs and their public key serves as their P2PWNC username. Users store in their portable devices, alongside their private key, *user certificates* that are signed by their home domain's private key. User certificates bind a username to a home domain name; their format is depicted in Fig. 1.

The user certificates encode only a very basic trust model. In the absence of a central authority and with the requirement for an open system, P2PWNC and its trust model is more similar to open ad hoc networks than to traditional roaming infrastructures where domains must know each other's identity and address beforehand. In P2PWNC (a) domains may not know of one another beforehand; (b) any entity may claim to be a new domain or a user of that domain simply by generating appropriately formatted names (i.e., keys) and certificates; and (b) domains could have an arbitrary number of identities. For our analysis we assume that domains do not trust each other, roaming users and visited domains do not trust each other, and home domains only trust their users for a limited time, which is encoded in the expiration date of the user certificate. We relax the definition of *successful authentication* to mean only that a visited domain has somehow established (a) that a roaming user is associated with a home domain and (b) that the user's home domain follows the reciprocity rule.

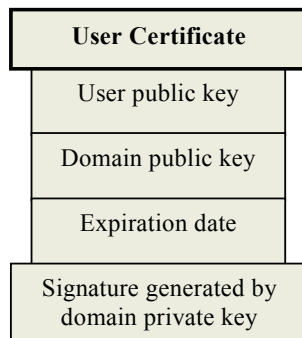


Fig. 1. P2PWNC user certificates. The public keys also serve as usernames.

2.2 Signed Receipts

When visiting users receive service from a visited domain they sign receipts that they give to the provider. Each receipt contains:

- ☐ a copy of the user's certificate;
- ☐ the name of the visited domain;
- ☐ a timestamp;
- ☐ service-specific information such as the volume of traffic that the visited domain routed for the roaming user;
- ☐ the user's signature, signing all the above.

A receipt encodes a transaction: the home domain represents the consumer and the visited domain represents the provider in this transaction. These receipts are public information and are used as input to the authentication algorithm that domains execute.

We also define the *Time-to-Live* (TTL), a universal system parameter that represents a time interval after which receipts expire.

All interested parties may verify that the user private key that corresponds to the user public key found in a receipt (as part of the user certificate) has signed the receipt. Also, they can verify that the domain private key that corresponds to the domain public key that is part of the user certificate has signed the user certificate. In essence, through one level of indirection (the user certificate), interested parties may verify the two domains that were involved in the transaction that a receipt encodes.

2.3 Disconnected Operation and Decentralized Storage

A central P2PWNC requirement discussed in [6] is that user authentication, as defined in Section 2.1, should proceed even if a user's home domain is offline. The certificates carried by a user are enough to prove his association with his home domain. However, the roaming user may not possess copies of all the receipts issued to his home domain, receipts that may be required to prove rule compliance. This is why copies of all receipts received by domains are stored in a decentralized storage subsystem where they remain available at least until they expire. For the following two sections, however, we can simply assume that all receipts received by domains are copied to an always-on logically centralized repository.

3 System Operation

3.1 Enforcing the Reciprocity Rule

So far, we have not really defined the 'reciprocity rule'. The model in [7] refers to a 'fixed-fee' type of rule. In P2PWNC, however, there is no straightforward way to check that a remote domain is constantly 'paying the fee', i.e., that it has an amount of bandwidth constantly reserved for visitors. In essence, we cannot check for availabil-

ity of bandwidth in a remote WLAN over the Internet without a trusted party at the remote WLAN that could verify this availability.

We therefore check for rule compliance indirectly: as long as there exist signed receipts stating that a specific domain provided service, we can assume that the domain did in fact provide service some time in the past (the exact time is part of the receipt). Three questions arise: (a) is the receipt ‘fresh’? (b) Do we trust the signature on it? (c) To what amount of resource consumption does the receipt correspond to?

The first question is a matter of deciding on an appropriate TTL for receipts (we discuss the TTL further in Section 4). An ideal TTL would be small enough to allow receipts to be garbage-collected quickly and in the process give incentives to participants to provide anew, and large enough to allow the system to remember good providers even if for some reason they happened not to provide to visitors during a specific time frame. The second question (trusting the signature) also is the subject of Section 4. The third question we discuss below.

3.2 Signing and Storing Receipts

When a roaming user appears in a visited domain, he presents his user certificate in order to identify himself and his home domain. The visited domain also presents its own name / public key. The public keys included in the certificates are used to establish wireless session keys and secure further communication against eavesdroppers and hijackers. (This form of mutual authentication is susceptible to man-in-the-middle (MITM) attacks, but the MITM attack is irrelevant in the context of the P2PWNC trust model, in which roaming users do not trust their providers, and would probably tunnel all their traffic to a trusted Internet proxy in any case.)

For the following, we assume that the providing domain is either altruistic or that the roaming user has been authenticated successfully (as defined in Section 2.1).

The P2PWNC WLAN transaction is as a series of post-pay exchanges. In the beginning, the visited domain routes a limited volume of traffic for the visiting user. Then, the visited domain presents a receipt for this traffic and asks the user to sign it. If the user declines, no further service consumption is allowed. To avoid persistently dishonest users, the domain tracks user and device identifiers and attempts to increase the cost of dishonest behavior by delaying or denying suspect login attempts. If the user does sign the receipt and the domain confirms the signature via the user’s public key, the user is allowed to continue.

At intervals specified by the visited DA, the domain presents new receipts and asks for the user’s signature. In the *service-specific* receipt field the domain may account for any resource whose consumption the user can confirm as well (in order to agree to sign it). If, for example, the domain accounts for the volume of routed traffic, this field’s value increases across receipts. All these receipts have the same timestamp (equal to the timestamp of the first receipt), a fact the user can confirm, and each one of these signed receipts makes the previous one obsolete: the visited domain will only store one receipt with a specific timestamp from a specific user, as other domains would not accept more than one such receipt during checks for rule compliance.

The visited domain stores all signed receipts in decentralized storage. For now, we can assume that this corresponds to a logically centralized, always-on server. Each domain stores a list of receipts it received on this server. Other domains can access

the list of receipts for any domain if they know the domain's name (public key) because the lists are placed in a hash table, keyed under a hash of the domain's name. This hash table is open for everyone to read. In Section 5 we distribute this hash table across domains. Expired receipts (according to the TTL) are deleted automatically.

4 Facing Sybil Attacks

4.1 The Problem and a Defence

The P2PWNC accounting mechanism described above is exposed to Sybil attacks [10]: an entity can generate any number of different domain identities since creating such identities only requires the generation of a key pair. The attacking entity can then generate any number of user certificates and associate them with any one of its domain identities. Then, the entity can use these user identities to sign fake receipts issued to one of its domain identities. Because receipts are used to check rule compliance, it would be trivial for an individual who does not even own a WLAN to consume without contributing: the individual need only create a fake identity for a home domain, issue a user certificate to himself, create any number of additional domain and user identities, and use these identities to sign receipts declaring that his (fake) home domain provided service to these identities. Because the system associates a Time-to-Live with receipts, the attacker would need to keep storing fresh receipts, but the cost of doing this may be significantly lower than operating a WLAN.

It is tempting to suggest that an analysis of the stored receipts by honest domains may alert them to suspicious patterns. This is indeed true. What we propose here, however, is that given the P2PWNC system model, *all* receipt patterns are suspect, except one. The objective of the authentication algorithm presented below is to detect this pattern, and only then to allow a P2PWNC visitor to login.

Consider a user c , associated with home domain C , visiting domain P (C is short for consumer, P for provider). In order to authenticate the user, P executes the breadth-first search algorithm below. We propose that this algorithm: (a) terminates eventually; (b) enforces the reciprocity rule; and (c) can result only in a very small percentage of unfair exclusions.

1. Add domain C to the `nodes_to_check` FIFO and to the `nodes_seen` set.
2. De-queue node from `nodes_to_check` and assign it to `current_node`.
3. Access the list of (live) receipts that show `current_node` as the providing domain. For every such receipt:
4. Check if the consuming domain on the receipt is in `nodes_seen`. If yes, get next receipt from list and repeat step 4. If consuming domain is P , `TERMINATE(SUCCESSFULLY)`. If not, add the consuming domain to the `nodes_seen` set and to the `nodes_to_check` queue.
5. If `nodes_to_check` is empty `TERMINATE(UNSUCCESSFULLY)` else go to 2.

This algorithm performs a breadth-first search on a tree of domains with C at the root. A successful termination of this algorithm is depicted in Fig. 2. This tree is a view of

the greater directed graph of (live) system receipts. In the graph and in the tree, the nodes represent domains and the edges represent receipts. Edges start from the consuming domain and point to the providing domain. An edge exists if there is at least one (live) receipt connecting the two domains.

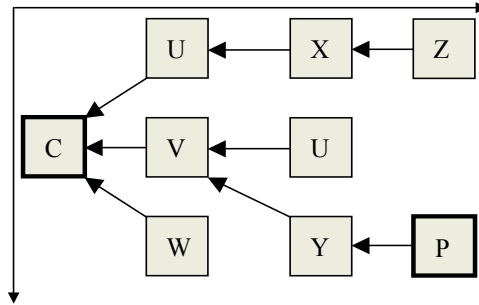


Fig. 2. A successful termination of the authentication algorithm. Domain P discovers itself in level 3 of the tree. Note that domain U appears more than once, but its child node, X , is accessed only once. The enclosing arrows depict the directions of the breadth-first search.

For simplifying the discussion below, assume that P and C are domains with only one registered user each, and that the identifiers P and C denote both the domain and its registered user, as meaning will be clear from context.

By following this algorithm, P attempts to find itself in a tree with C at the root and terminates successfully only if it succeeds in doing so. If P is found at the first level of this tree (the root is at level zero), then this means that P has (recently) received service from C . In such a case, according to the reciprocity rule, P should now provide service to C . If P is found at the second level of this tree, then this means that P has (recently) received service from somebody who (recently) received service from C , and so on. We propose that if this algorithm terminates successfully then there exists at least one real domain in the path connecting P to C (in addition to P) because P has signed a receipt for consuming resources, and assuming P trusts its roaming registered users, such a receipt can only be signed when a (real) visited domain provides resources to P . The identity of the domain that provided to P (Y in the example of Fig. 2) is irrelevant: what is important is that a domain that called itself Y provided service to P . Also, according to the stored receipts, V provided service to Y and C provided service to V . It would be in accordance with the reciprocity rule, now, for P to provide service to C . Even if V , Y , and C were all aliases of the same domain, which also means that the receipts connecting them are fake, P can trust that the last receipt, the one signed by P itself, is real, which means that P had received service from the $V/Y/C$ domain and now has an opportunity to pay back.

If the algorithm terminates unsuccessfully, P exhausted the (live) receipts without discovering itself anywhere in the tree. This is bound to occur if there are a finite number of domain identities because the algorithm does not visit a domain identity more than once when searching for receipts. If P does not detect itself, *all* the domains seen in the tree could be fake identities created by C and all the edges fake receipts signed by these fake identities, in an attempt by C to fool the reciprocity mechanism, so C should be excluded in order for the right incentives to be provided.

On the other hand, by actually detecting itself in the tree, P is not concerned about potential Sybil attacks any more. P only wants to uphold the reciprocity rule: P was indeed offered service by the community, so P now gives back to the community. After contributing, however, P must mark the particular receipt that connected him to C to be ‘used’. If P did not do this, a single act of consumption on P ’s part could prompt many contribution acts. This way, even if the domain that actually provided service to P colludes with other domains, the colluding team as a whole will not be able to take advantage of a single act of contribution more than once.

4.2 Evaluation

What is the percentage of unfair exclusions that this algorithm causes? To measure it we simulated a P2PWNC community that used the above algorithm for authentication. In our simulation, all unsuccessful terminations of the algorithm were unfair exclusions because none of the simulated domains was attempting to free-ride. (The algorithm would detect such a free riding domain immediately since no receipts proving contribution would exist and the algorithm would terminate in its initial steps.)

In order to choose an appropriate simulation model, we support that a metropolitan-sized P2PWNC would most probably exhibit a *small world* [9, 11] phenomenon³. More specifically, we base our P2PWNC model on a small world model similar to the one presented in [9]. Our set of nodes representing P2PWNC hotspots are identified with the set of lattice points in a $n \times n$ square (see Fig. 3). For a universal constant p all nodes have *local relationships* with nodes within lattice distance p . Each node also has q *distant relationships* with nodes chosen uniformly at random during system initialization. This models relationships that could also cause interactions, and, by extension, P2PWNC transactions.

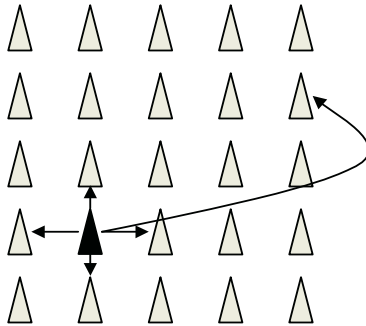


Fig. 3. A P2PWNC with 25 hotspots ($n = 5$). Here, $p = 1$ and $q = 1$. We focus on the relationships of the black hotspot, which we depict using arrows that start from it.

In our simulations, all hotspots are symmetric, with exactly one registered user. For a given set of *movement probabilities* encoded in the universal constants p_l , p_d , and

³ Small world graphs mainly arise in social contexts but have also been used to model user ad hoc networks [12]. In running our simulations, we arrived at the same conclusions as [12] and [13], namely that due to random mobility the receipt graph is even more connected.

p_r (all in the range $[0, 1]$), these registered users spend (a) p_l of their time accessing local contacts (choosing uniformly at random which one), (b) p_d of their time accessing one of the q distant contacts (choosing uniformly at random which one), and (c) p_r of their time accessing any hotspot chosen uniformly at random. This last probability of movement acknowledges the fact that completely random movement is also a possibility.

To allow for the first receipts to be created, a percentage of the hotspots (p_a , in the range $[0, 1]$) are altruists at the beginning of time. Altruistic hotspots (as defined in Section 2.1) authorize users without running the authentication algorithm. Nevertheless, they do ask for signed receipts. For the simulation, we also apply the following heuristic: every altruistic hotspot stops being altruistic and becomes a regular non-altruistic (i.e., rule-enforcing) hotspot when the hotspot's registered user successfully passes his first authentication in a non-altruistic hotspot. There are various local policies that hotspots may use to decide something similar. What is important is that new hotspots must start by being altruistic in order to collect a certain number of receipts and that they should eventually turn to non-altruistic hotspots in order to provide correct incentives to visitors and stop encouraging free-riders.

The main result of our simulations is that for realistic sets of relationships and mobility patterns there is always a TTL value *above* which unfair exclusions can drop below 1% of the total number of authentications, and *below* which the system eventually collapses (i.e., it surpasses 99% unfair exclusions). We observed that whenever the unfair exclusion rate shows even the slowest, but persistent, rate of increase, then the system is bound to collapse. This can be explained if we consider that as altruist numbers are reduced according to the heuristic, rule-enforcing providers increase. If an unfair exclusion due to expired receipts occurs then this would result in yet another receipt not being created. As time progresses, the total number of receipts in the system decreases and eventually we return to the initial conditions, i.e., with no receipts in the system, but without altruists anymore all visits result in exclusions.

We simulated 1024 hotspots arranged in a 32×32 grid with $p = 1$, $q = 2$, $p_l = 0.05$, $p_d = 0.05$, and $p_a = 0.99$. In the following figures (4 through 6), we depict results from simulations with four different sets of parameters: two different values for the TTL (30 and 60) and two different values for p_r (0.00 and 0.02), with the remaining parameter values as above. For every step of the simulation, all users visit foreign hotspots according to the preset relationships and probabilities of movement. With the parameter values above, users stay at home 90% or 88% of the time (if $p_r = 0.00$ or $p_r = 0.02$ respectively). The TTL is measured in simulation steps.

Fig. 4 depicts the evolution of altruistic hotspots. Altruistic hotspots in all tests start at 99%. Random user movement (with $p_r = 0.02$) across all hotspots causes our heuristic to turn altruistic hotspots to rule-enforcing faster because hotspots have a chance of interacting with users from a greater number of foreign hotspots which gives more chances to our heuristic test to run and eventually succeed.

Fig. 5 supports our main finding, namely that there exists a TTL value above which there would be insignificant gains in performance (low percentage of unfair exclusions) since unfair exclusions are very near zero, and below which the system eventually collapses (unfair exclusions reach 100% of authentications when, eventually, all receipts expire and new ones cannot be issued because altruists have disappeared). Note that random movement is not required to achieve low unfair exclusion rates. Of course, in the case of system collapse, random movement can delay the collapse

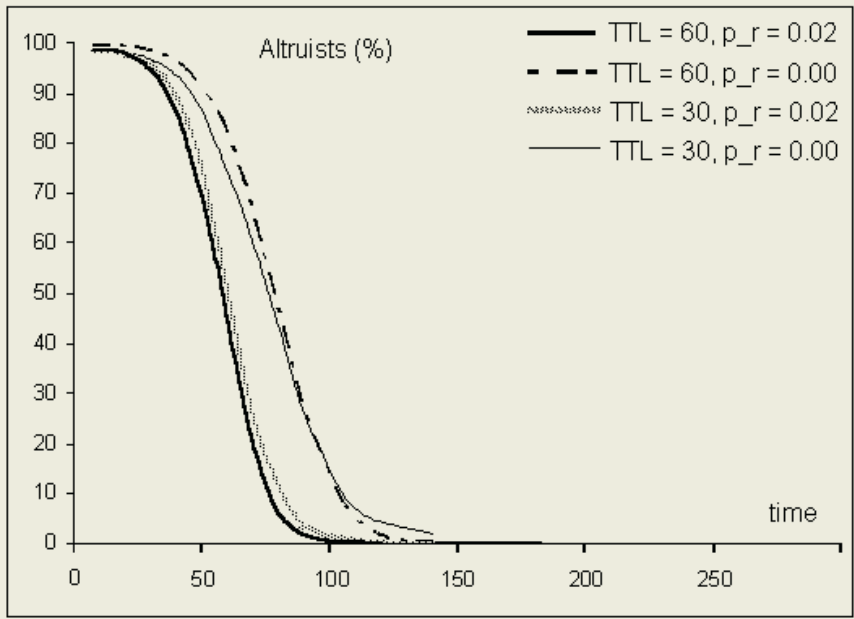


Fig. 4. The percentage of altruistic hotspots.

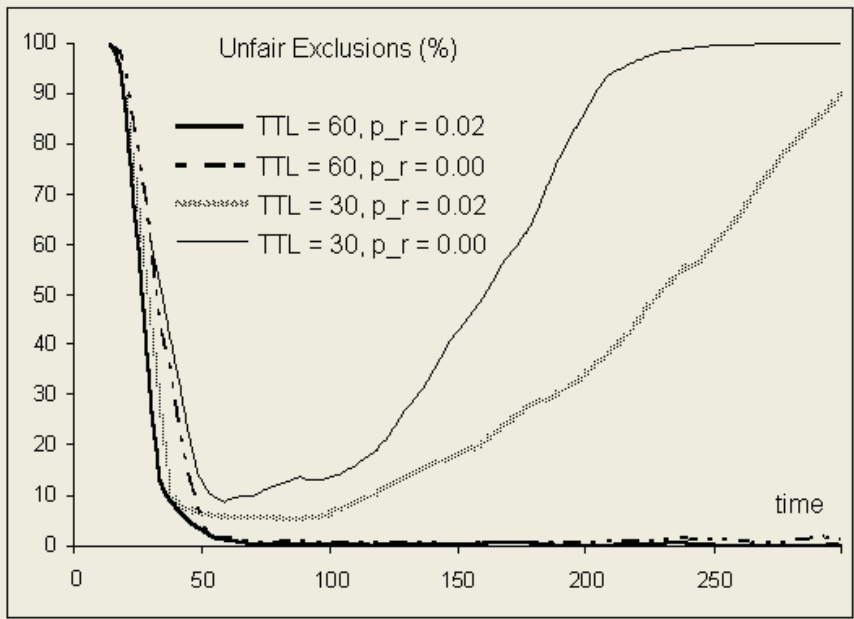


Fig. 5. The percentage of unfair exclusions.

somewhat (as can be seen in Fig. 5). In the real world, collapse can be avoided and unfair exclusions can be even lower if some hotspots remain altruistic (here, after the 150th simulation step, approximately all hotspots have turned to rule-enforcing hotspots – see Fig. 4). In fact, a reverse heuristic could be that after an unfair exclusion a rule-enforcing hotspot could turn altruistic, and then the original heuristic would apply again. (Also, in the real world, all new hotspots must be altruistic when they first enter the system, as we mentioned before.)

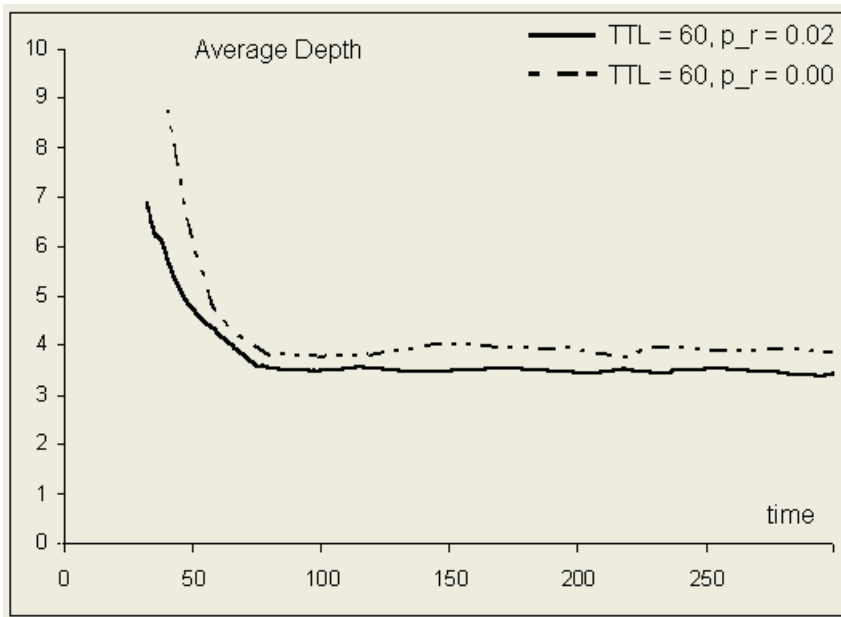


Fig. 6. The average tree depth at which a successful authentication terminates.

Fig. 6 is a direct result of our small-world model: in the two sustainable 1024 hotspot systems (those with $TTL = 60$) it takes only four steps on the average in the receipt graph to reach a hotspot from any other hotspot. Also, it takes slightly less than four on the average if there is some element of completely random movement.

4.3 Enhancements

Even though the algorithm appears to work theoretically there is still the practical issue of efficiency. Given that receipts are stored across different nodes in decentralized storage, it could take a long time for the authentication algorithm to access them before it terminates, which is a significant cost that the potential provider may not have an incentive to pay. One way to reduce this cost is to assist potential providers by *pre-computing* paths. More specifically, the domain agent in every P2PWNC domain can keep a background process running that searches the decentralized storage for new receipts. The goal is to build a tree of receipts (like the one in Fig. 2), with the domain in question at the root of this tree, and to update it continually as receipts expire and

as new receipts enter the system. The home domain could send fresh versions of this tree to its roaming registered users whenever the opportunity arose. The tree does not have to contain the receipts themselves. Instead, for every tree node, needed are only some of the least significant bits of the domain name that corresponds to this node.

The procedure above concerned domains that played the part of consumers. Providers, on the other hand, can also maintain a similar tree. These ‘provider’ trees would have one important difference: the root would again be the domain in question but the directed edges would point to the opposite direction.

Using standard probability computations, we can see that the probability that the provider and the consumer would have at least one node in common increases rapidly even with modest increases in the number of nodes in these trees, and even for very large P2PWNCs. A node chosen by both provider and consumer is equivalent to saying that at least one connecting path would exist connecting the two. The provider would only have to check the receipts on this path (also, unlike the mobile client, the provider can save the entire receipts, not just the least significant bits of names).

5 Decentralized Storage Subsystem

In order to build a self-sufficient receipt storage system that relies only on the domain agents themselves, we borrow ideas from the *Chord* [14] peer-to-peer lookup protocol. The P2PWNC Public Key Infrastructure we proposed is compatible with the one proposed in [15], where it Chord security extensions are presented and where it is suggested that each live node of the storage subsystem uses as *secure Chord identifier* the hash of a data structure that includes the node's public key, and its current IP address. Our main concern is the *availability* of receipts, as storage nodes join and leave the system. P2PWNC receipts are self-certifying data structures, so no additional protection is needed to ensure their integrity. Inbuilt Chord securities (such as *consistent hashing*) and the PKI extensions described in [15] ensure that an adversary cannot affect a large part of the system without first controlling a significant number of IP addresses in different subnets. In order to avoid free riders at the storage layer, nodes need to process queries only when they are coming from nodes that provably store their share of receipts according to the Chord mapping function. This is relatively easy to check at the Chord layer in order to exclude nodes that refuse to take on their share of the load.

6 Conclusions

In this paper we claimed that a metropolitan-sized P2PWNC would exhibit a small world phenomenon and with the help of simulations we showed how the fundamental Sybil attack problem can be circumvented. Our goal was to ensure that participants that attempted to free ride would be excluded from consuming community resources. We linked system memory (expressed as the receipts' Time-to-Live) to system performance (expressed as the percentage of unfair terminations of the P2PWNC authentication algorithm). We showed that there exists a TTL value above which there

would be no appreciable gain in system performance. We also presented our ideas on how to make the basic authentication algorithm more efficient and we outlined the foundations of the P2PWNC decentralized storage subsystem.

7 Acknowledgments

This work is partly supported by the EU research project ‘Market Management of Peer-to-Peer Services’ (MMAPPS, RTD No IST-2001-34201). The authors gratefully acknowledge ongoing discussions with their MMAPPS partners, and particularly Panayotis Antoniadis and Thanasis Papaioannou.

References

1. Nokia – Nokia 9500 communicator. <http://www.nokia.com/nokia/0,,54106,00.html>.
2. GSM Association. PDR IR.61, WLAN roaming guidelines, 2003.
3. Wireless Broadband Alliance. <http://www.wirelessbroadbandalliance.com>.
4. Boingo Wireless Inc. <http://www.boingo.com>.
5. iPass Inc. <http://www.ipass.com>.
6. E. C. Efstathiou and G. C. Polyzos. A peer-to-peer approach to wireless LAN roaming. 1st ACM Int’l Workshop on Mobile Applications and Services on WLAN Hotspots, 2003.
7. C. Courcoubetis and R. Weber. Asymptotics for provisioning problems of peering wireless LANs with a large number of participants. In Proc. WiOpt’04, UK, 2004.
8. E. Adar and B.A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), 2000.
9. J. Kleinberg. The small world phenomenon: an algorithmic perspective. In Proc. STOC’00: 32nd ACM Symposium on the Theory of Computing, Portland, OR, 2000.
10. J.R. Douceur. The Sybil attack. 1st Int’l Workshop on Peer-to-Peer Systems, 2002.
11. S. Capkun, L. Buttyan, and J.-P. Hubaux. Small worlds in security systems: an analysis of the PGP certificate graph. New Security Paradigms Workshop, 2002.
12. S. Capkun, L. Buttyan, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1), 2003.
13. S. Capkun, J.-P. Hubaux, and L. Buttyan. Mobility helps security in ad hoc networks. 4th ACM Symposium on Mobile Ad Hoc Networking and Computing, 2003.
14. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, vol. 11, Feb. 2003.
15. E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. 1st Int’l Workshop on Peer-to-Peer Systems, 2003.

Mobile Qualified Electronic Signatures and Certification on Demand

Heiko Rossnagel

Chair of Mobile Commerce and Multilateral Security,
Johann Wolfgang Goethe University Frankfurt, Gräfrstr. 78,
60054 Frankfurt, Germany
heiko.rossnagel@m-lehrstuhl.de
<http://www.m-lehrstuhl.de>

Abstract. Despite a legal framework being in place for several years, the market share of qualified electronic signatures is disappointingly low. Mobile Signatures provide a new and promising opportunity for the deployment of an infrastructure for qualified electronic signatures. We analyzed two possible signing approaches (server based and client based signatures) and conclude that SIM-based signatures are the most secure and convenient solution. However, using the SIM-card as a secure signature creation device (SSCD) raises new challenges, because it would contain the user's private key as well as the subscriber identification. Combining both functions in one card raises the question who will have the control over the keys and certificates. We propose a protocol called Certification on Demand (COD) that separates certification services from subscriber identification information and allows consumers to choose their appropriate certification services and service providers based on their needs. We also present some of the constraints that still have to be addressed before qualified mobile signatures are possible.

1 Introduction

In the directive 1999/93/EC of the European Parliament and of the Council [ECDIr1999] legal requirements for a common introduction of electronic signatures in Europe were enacted. The directive sets a framework of requirements for security of technology used for electronic signatures. Based on certificates issued by certification authorities, which certify public keys for a person registered by a registration authority, electronic signatures can be created with a so-called "secure signature creation device" (SSCD), carrying the private keys of a person.

The EC-directive distinguishes between "electronic signatures" and "advanced electronic signatures" [ECDIr1999]. An advanced electronic signature is defined as an electronic signature that meets the following requirements:

- “(a) it is uniquely linked to the signatory;
- (b) it is capable of identifying the signatory;

(c) it is created using means that the signatory can maintain under his sole control; and

(d) it is linked to the data to which it relates in such a manner that any subsequent change of the data is detectable;” [ECDir1999]

Certification Service Providers can issue certificates for advanced signatures that will be qualified if they meet the requirements of Annex I of the directive. Those advanced signatures with qualified certificates will be referred to in this paper as qualified signatures.

In Germany and Austria, the local implementation of the EC directive requires evaluation of the SSCD to be done against ITSEC E4 or CC EAL 4+ levels [FuFr2000]. For directory services, stringent 24/7 availability and durability is required. Revocation lists and other feasible technology must be available to all accepting parties of signed documents. The EU suggests the implementation of a public evaluation infrastructure under control of a government authority. Germany has already implemented a system of evaluation service companies, evaluation consulting companies and the Regulatory Authority for Telecommunications [RegTP2004] as the responsible government authority.

The deployment of signature card products focused so far on smart cards with evaluation against the requirements for lawful electronic signatures. Based on these, personal computer based signature applications have entered the market. These applications require smart card readers attached to the workstation, thereby preventing user mobility.

The market share of EC-directive conforming smart cards is disappointingly low, failing to meet any involved party’s expectations. This has partly been blamed on the incompatibility and missing standards of existing products. Also the lack of customers prevents companies from investing in signature products. As a result almost no commercial usage for qualified electronic signatures exists. Consequently no customers seek to obtain signature products.

There are numerous activities trying to enlarge the potential consumer base like putting key pairs on national identity cards [FSEID2004]. Lately there have been some efforts towards mobile signatures [ETSI] [Raddic2004] and this approach might have a chance to break up the deadlock of missing customers and missing applications. However, there are numerous problems to be solved, before qualified signatures can be created with a mobile device.

The first part of this paper (Section 2) gives an overview on the possible approaches for mobile signatures especially focused on SIM¹-based signatures and its challenges in detail (section 3). In section 4 we present a protocol for SIM-card deployment solving most of these challenges. Section 5 provides an outlook on possible usage scenarios of that protocol and section 6 focuses on the security of mobile devices. In section 7 we examine special constraints of mobile signatures and section 8 concludes our findings.

¹ Subscriber Identity Module

2 Mobile Signatures

Mobile signatures are electronic signatures which are created using a mobile device and rely on signature or certification services in a location independent telecommunication environment. They allow signatory mobility beyond fixed, secure desktop workstation with trusted, personal signing equipment [FrRaRo2003]. Although using mobile devices for signature creation has several shortcomings (e.g. display size, communication costs, limited computing power), the high market penetration of cell phones [GSM2004] and the mobility gained make this effort potentially successful and promising.

Two possible signing approaches in the mobile environment have been proposed in the past: signatures created in centralized signing server environments located at service providers like mobile network carriers; and electronic signatures created inside the signer's mobile device using a smart card.

2.1 Server Based Electronic Signatures

Server based electronic signatures are signatures created by a service provider for a specific customer. Figure 1 illustrates such a server infrastructure.

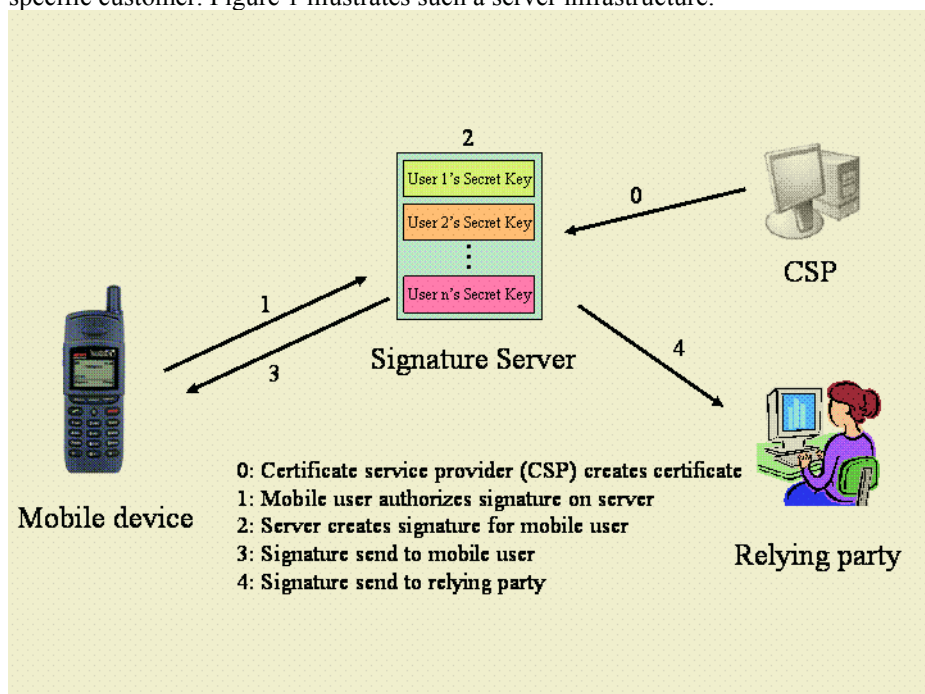


Fig. 1: Server based electronic signature infrastructure

With server based signatures it is important to distinguish between signatures that have a corresponding certificate issued under the name of the customer and signatures

with certificates issued under the name of the service provider or an employee of this provider.

In the first case it is necessary that the customer transfers his private key to the service provider. However, according to Art.2, 2(c) the signature has to be created by means that the signatory can maintain under his sole control to achieve the status of an advanced signature [ECDir1999]. By giving away his private key this premise can not be fulfilled [FrRaRo2003]. In the case of signatures whose certificates are issued under the name of the service provider you can not assume these signatures to be legal signatures of the customer. They are signatures of the signature service provider and only enable an identification of the provider. Those signatures can achieve the status of advanced signatures with qualified certificates as long as they fulfill the requirements of Annex I and are provided by certification service provider who fulfills the requirements of Annex II. Therefore, the signature service provider acts as a replacement for the customer. However, based on the signature of the provider it can not be verified that the customer really authorized the signature. Neither the integrity nor the fact that the user authorized it can be proven. There are possible technical solutions to accomplish the integrity and accountability of his authorization but they would require a security environment on mobile devices that would enable the device to create qualified signatures itself [RaFrRo2003].

2.2 Client Based Electronic Signatures

Signatures can be created inside the mobile device using a secure signature creation device which has to fulfill the requirements of Annex III. Using a multiple smart card solution, the signature smart card, certified by a certification provider, is inserted into the mobile device which already contains the usual SIM-card. Therefore, the signature process takes place on the mobile device and the user is able to use basically any signature card available on the market. This can be achieved by either exchanging the SIM-card with the signature card (Dual Chip) or by having an additional chip card reader within the mobile device (Dual Slot). The first solution is very inconvenient for the signatory since he has to switch of the phone to exchange the cards for the signature creation and again to use the phone functionality. In the latter case a specialized mobile phone is required that has multiple smart card slots which almost none of the current mobile phones do.

It would also be possible to use a single smart card that contains the SIM telephone functions, as well as the secure signature creation device. This can be achieved either by leaving some free space on the SIM-card, on which the components of the signature creation device can be installed later on, or by shipping SIM-cards with pre-installed signature functionality that has to be initialized and activated.

We propose the usage of evaluated smart cards suitable for qualified electronic signatures which are extended by the SIM functionality and usable through a unified interface, e.g. with the USIM² specification TS 21.111 [3GPPSpec]. Another approach might be the migration and evaluation of USIM with a full WAP³/WIM⁴ implementa-

² Universal Subscriber Identity Module

³ Wireless Application Protocol

tion for the purpose of lawful mobile signing [WAPF2004]. Evaluation must be carried out with ITSEC or Common Criteria within an evaluation process similar to the evaluation summarized in [FuFr2000].

3 Challenges of SIM Based Signatures

Using a single smart card for both functionalities provides the most convenient solution for the signatory. He can sign documents and distribute them via communication services of his cell phone like GPRS⁵ or UMTS⁶. To ensure that the requirements of Art.2 2(c) are met, it is necessary to provide some sort of reliable access control to the signature functions. The usual PIN used to control the access to the telephone functions is not sufficient, since users can keep their phones and SIMs unlocked for convenience. Like traditional signature cards, SIM-cards can be certified according to security evaluation criteria and are under control of the user.

However, using a single smart card for multiple purposes raises new questions and challenges. The SIM-card is issued by the telecommunication provider, while the SSCD is issued by a certification service provider. Combining both functions in one card raises the question who will have the control over the keys and certificates.

The simple solution is that the deploying carrier also initializes the signature secrets to act as a trust provider for their customers. This seems to be reasonable at first glance, since some of the european carriers already own and maintain trust centers (i.e. Deutsche Telekom), but there are several shortcomings, which make this approach unpractical.

First of all the customer wants to leave the store with his SIM-card right away, so he can use his mobile phone instead of waiting several weeks for the certification process to be completed. Furthermore, binding the keys to a carrier creates a great hindrance for the customer to switch to a cheaper carrier in the future. From the carriers point of view this would of course be a positive effect. From the customer's perspective, however, it would be much better to be able to choose freely between different certification service providers.

Also due to the lack of success of the signature market so far most providers probably do not want to invest in building and maintaining their own trust center to provide certification services. In addition, they don't want to change their distribution channels unless they expect an increase in revenue.

Therefore, a different solution for mobile signing and certification is needed, that allows separation of subscriber information and certification services.

⁴ Wireless Identity Module

⁵ General Packet Radio System

⁶ Universal Mobile Telecommunication System

4 Certification on Demand

The mobile operator could sell SIM-cards equipped with a key generator for one or more key pair(s) which can be used for the signing functionality. After obtaining the SIM-card from the mobile operator, the customer can then generate the keys and activate the signature component and the public key(s) can be certified by any Certification Service Provider on demand.

Through the separation of the telephone functionality and the (possibly later) certification of the user's identity by a certification service provider, both functions can be sold separately and can be obtained from different providers.

The carrier will probably face increased costs for the signature capable SIM-card but can also expect increasing traffic caused by signature services. All distribution channels will remain unchanged.

Figure 2 illustrates the necessary steps for the distribution of the SIM-card and the certification process.

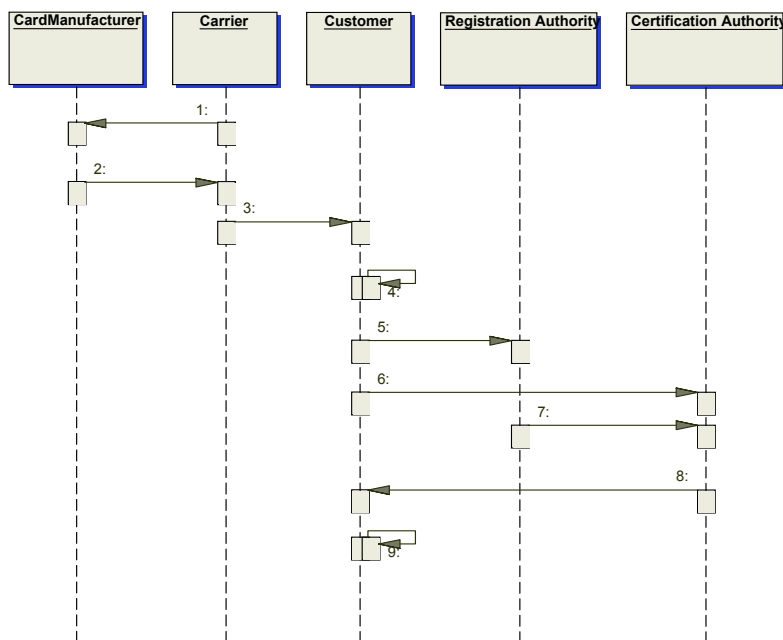


Fig. 2: Certification on Demand Protocol

1. The carrier gives his IMSI⁷/Ki⁸ pairs to a card manufacturer.
2. The card manufacturer returns a SIM card containing an IMSI/Ki pair, a key generator for the signature application and the public key of the RootCA to the carrier.

⁷ International Mobile Subscriber Identity

⁸ Individual subscriber authentication key

3. The SIM card is sold to the customer and the carrier provides a nullpin that is used to generate the keys and activate the signing functionality.
4. The customer generates the keys and activates the signing functionality by entering the nullpin.
5. The customer registers at a Registration Authority of his choice, providing identification information and his public key.
6. The customer sends his identification information signed with his private key over the air to the Certification Authority.
7. The Registration Authority sends the public key and the identification information to the Certification Authority.
8. If the information provided by the customer and the Registration Authority match, the Certification Authority issues a certificate for the customer and sends it over the air to his mobile phone.
9. The user can verify the validity of his certificate by checking the certificate issued by the RootCA of the Certification Service Provider.

This protocol makes no changes to the existing distribution infrastructure of mobile operators. The steps 1 to 3 remain the same way they used to be before, apart from the fact that the card manufacturer puts additional information and functionality (signature key generator, public key of RootCA) on the SIM card. In order to ensure that the card manufacturer does not know the private key of the user the key generation should be done by the card. The customer is not forced to certify his keys and can use the SIM for telephone functionality only. He could also activate the signing functionality without going through the certification process for example as a security token. If he wants to be able to make legal binding electronic signatures, he has to go through the complete process to obtain a qualified certificate. He can do this by freely choosing the CSP.

The nullpin to generate the keys and activate the signing functionality in step 4 is used to ensure that no signatures can be created before the customer has control over the SIM card. If the signature application has been activated before, the user will recognize this when entering the nullpin.

Step 6 could be omitted but serves as insurance for the customer to ensure him that the integrity of his identification information will be preserved.

If the customer wants to change his CSP, he only has to repeat steps 5 to 9 with his new CSP. If the customer wants to change his carrier, he has to go through the whole protocol again, but can register with his current Certification Service Provider.

5 Possible Applications

5.1 Enabling Security Infrastructures

There is a need of corporations to provide their mobile workforce with secure access to the corporate backend. So far security tokens have been used to allow this functionality. These tokens are expensive and stored on extra hardware that needs to be carried around and can easily be lost. Putting these credentials on a SIM, that will be

placed in the mobile phone, reduces the risk of losing the credential as well as the costs. But some corporations greatly object to leave their private keys and certificates in the hands of their mobile operator.

With Certification on Demand the corporation's IT security department can obtain COD enabled SIMs from the corporation's cellular contractor and initializes them for the corporate mobile security infrastructure. The WiTness project [WiTness] sponsored by the European Union implements such an infrastructure.

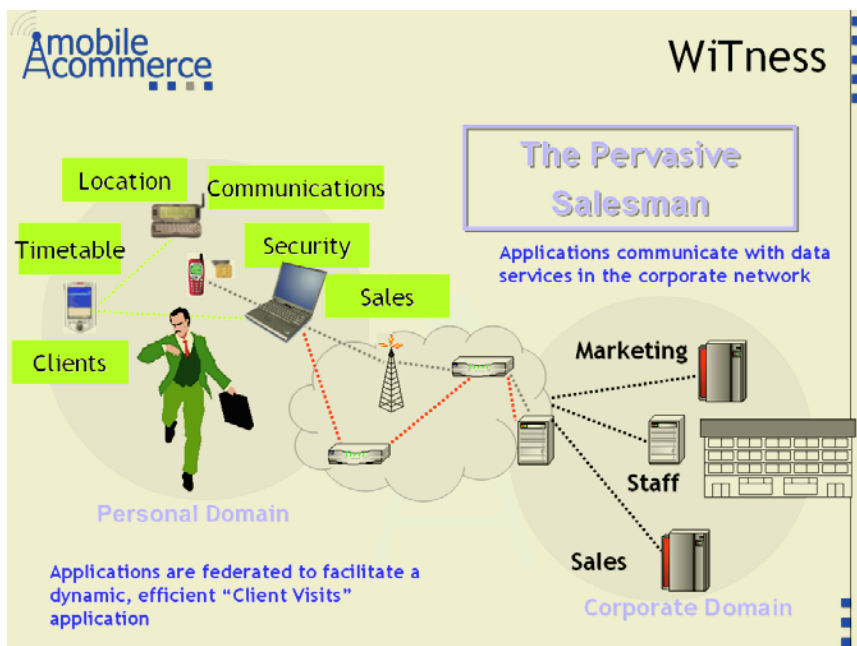


Fig. 3: WiTness Pervasive Salesman Scenario [WiTness]

Figure 3 shows an application scenario where a “pervasive salesman” has secure, corporate-controlled access to all data available to him in the corporate information system. Access is controlled by a security module based on a SIM with additional security functionality.

5.2 Multilateral Secure Financial Transactions

Storing bank credentials on a SIM may help the migration from plastic to mobile phones. A COD infrastructure allows financial institutions to certify and enable mobile subscribers to use banking services online through their mobile terminal and SIM. Credentials could be certified by the bank itself, like the credentials used on bank cards. Therefore, the bank can still have the control over the credentials while the mobile operator still can issue the SIM cards without giving their IMSI/Ki pairs away to the bank. This would enable the bank to offer services like transactions, brokerage or checking the account balance based on the credentials stored in the SIM.

This functionality can and has been realized without the Certification on Demand protocol but only if the banks and carriers are willing to cooperate. In the Czech Republic T-Mobile [TMO2004] and the Czech banks agreed to send their critical information to Giesecke&Devrient, a card manufacturer who started producing banking enabled SIMs [GuD2004]. However, the COD protocol would enable banks to use SIMs as credentials without having a contract with the mobile operator.

5.3 Enabling Mobile Electronic Consent and Identity Management

Many mobility applications rely on a user's consent towards reducing his privacy for a particular service. Examples are location based services on cellular networks, situation based marketing scenarios and tracking technology following users to support them with information they need in-time and in-place. A secure provable electronic consent of users can be achieved using electronic signatures on SIM-created credentials that may contain information about time, intent and recipient of the electronic consent. Research has found SIMs to be on the edge of a global identity management infrastructure [Rann2003]. In the near future, personal or role attributes customized for particular application areas (e.g. online dating, identity management) could be managed on SIMs on demand from their owners.

5.4 Using COD in Deployment of Electronic Identity Cards

If the signature credentials are stored on an identity card issued by the government, the same problems as described in section 3 occur. The Government has to issue the identity card but does not want to act as a certification service provider. Using the COD or a similar protocol enables current CSPs to certify the keys of the recipients of the identity cards.

6 Trusted Mobile Devices

The mobile device serves as the card reader, storage device for the document to be signed and as a display for the signature application. Therefore, it must be ensured that the data shown on the display is identical with the data signed by the signature card. This is commonly known as "What You See Is What You Sign" (WYSIWYS). The operating system used on the mobile device has thus a pivotal importance to ensure the integrity and accountability of the electronic signature.

If the authorization mechanisms, memory protection, process generation and separation or protection of files in an operating system are flawed, an attacker may gain access to the different internal processes. He might take advantage of this situation to generate forged signatures.

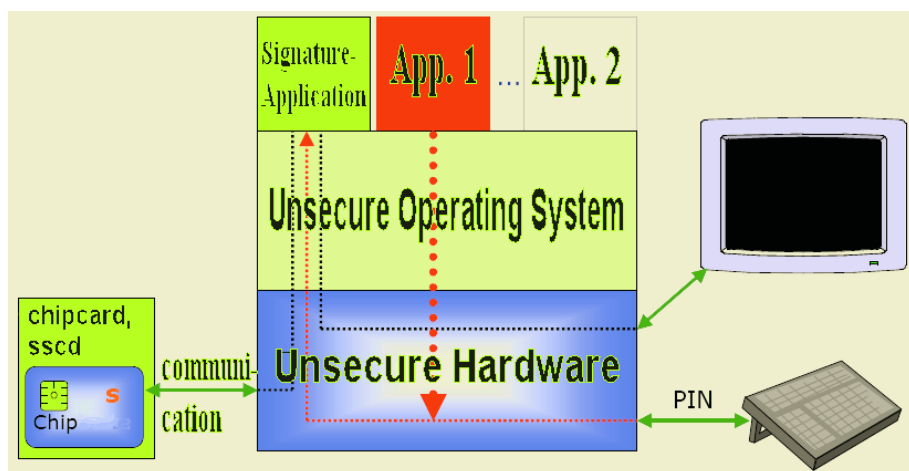


Fig. 4: Manipulated digital signature [Federr2003]

Figure 4 illustrates that application 2 as a malicious program can for example intercept the PIN. An even considerably higher risk exists, however, if the malicious application changes the data to be signed after they are displayed to the user. Due to the virtual unrestricted hardware access, a malicious program is able to manipulate all data transmitted to the signature application before the actual signature takes place.

In the past mobile phones were “closed” devices that gave the user no possibility of installing programs or storing data apart from address book entries. But with increasing computing power and storage capabilities, new and open operating systems like PocketPC [Pocket2004] and Symbian [Symbian2004] were developed, which allow users to install any program they like. This of course raises the possibility that malware or Trojan horses are installed by the user or a third person.

Although a tamper resistant mobile phone could be build and certified most of the features of present phones would probably not be available for this phone. Therefore, it will probably fail to get a high market penetration. The only solution that seems to be promising is to have a small microkernel as a secure platform which runs the signature application and an additional common operating system running on top of the security kernel.

The “Open Source” project at Saarland University Perseus develops such a system [Perseu2004]. It provides a small microkernel as a secure platform. The microkernel is responsible for the administration of the device, files, memory and processes and is loaded directly after booting. It is aimed at protecting security-critical applications by isolating the individual processes from each other. Perseus is based on the approach that a normal operating system runs like an application, and therefore the Perseus kernel lies below the operating system in the layer architecture. Only by being embedded below the operating system, which is still needed for ordinary applications, Perseus can permit isolated processes to take place system-wide between the applications. Isolated processes are not possible for applications within the standard operating system, however, but only between the individual “secure applications” and the Perseus operating system.

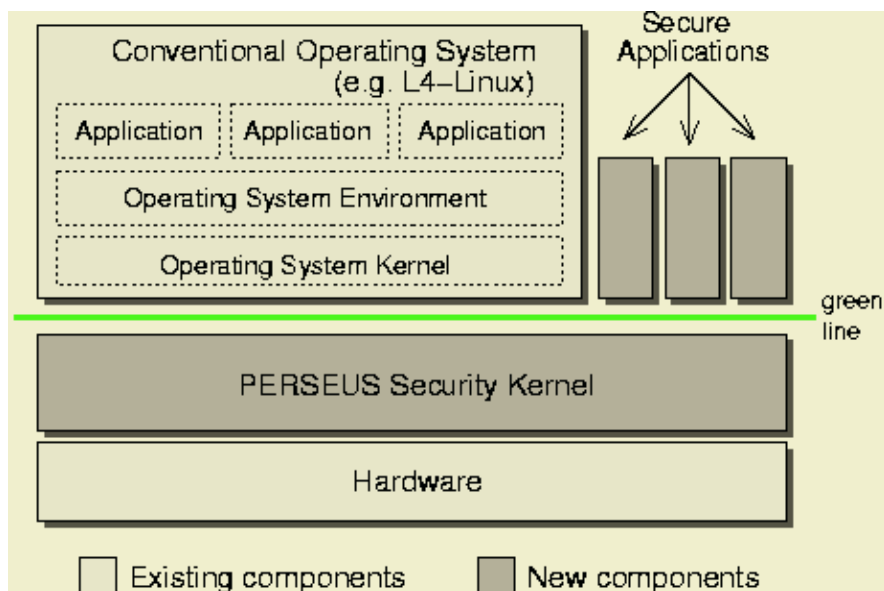


Fig. 5: System Architecture Perseus [Perseu2004]

In the Perseus prototype, the trustworthy user interface reserves a line in the upper section of the screen that is permanently under the control of the security kernel.

As the line or LED is under the sole control of Perseus, it cannot be misused by a compromised operating system. If the display indicates that the user is communicating with the Perseus kernel, the control of the full display and keyboard solely lies with the security kernel.

7 Mobility and Signing

Mobile signatures are made with mobile devices and therefore constraints have to be addressed that are not present in traditional signing infrastructures.

7.1 Data Transfer

First of all any traffic that is necessary will be accounted to the customer's bill. Therefore, it is essential to create as little data traffic as possible in order to get the customer to accept the additional costs. In the case of the signature creation, traffic is only necessary for the download of the document to be signed, if at all. In the process of signature verification, several documents, especially the keys of all CA's involved have to be downloaded in order to ensure the integrity of the verification process.

Revocation lists are a particular concern that has to be met. In order to be up to date with actual revocation lists the customer has to be "online" to be able to get access to the actual status of all the involved signatures and certificates. This could lead to lots

of data being transferred and a lot of additional costs. Standards like ISIS-MailTrusT [ISISMTT] can be useful as well as concepts of server centric support in document verification [Fritsch2002].

It would also be possible for the CSP to sponsor the additional data traffic in order to get customers to accept and use mobile signatures [FSMR 2003].

7.2 Storage

Mobile devices usually have a rather fixed amount of storage space. This is even more relevant, if one has to store the data on the SIM-card itself, for whatever reason possible. Therefore, the mobile signature application should whenever possible try to store the necessary information on a server of the service provider. This of course is in contrast to the goal of minimizing the necessary traffic for signature applications. Therefore, a trade off between cached information and information to be transferred has to be found. This is particularly important for the storage of root certificates, certification chains and certificate revocation lists for offline-verification. This problem might be solved by increasing storage space on mobile devices and the ability of modern devices to use external storage like SDCards.

8 Conclusion

Mobile Signatures are a promising approach to break the deadlock between missing customers and missing applications. The high market penetration of mobile phones enables certification service providers to target millions of potential customers. We analyzed two possible signing approaches (server based and client based signatures) and conclude that SIM-based signatures are the most secure and convenient solution. However, using the SIM as an SSCD seems to force the mobile operator to act as a trust provider and therefore to challenge the existing CSPs in a market that hasn't been successful so far. We proposed a protocol called Certification on Demand that separates subscriber information from certification services and therefore enables both industries to cooperate instead of compete with each other.

We also provided possible application scenarios that can be realized with Certification on Demand. But even with the certification problem solved, there are still a lot of open issues that have to be addressed before mobile qualified electronic signatures will be able to get market acceptance.

References

- [3GPPSpec] Specification of GSM, <http://www.3gpp.org/ftp/Specs/archive/>
- [ECDir1999] European Union: DIRECTIVE 1999/93/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 13 December 1999 on a Community framework for electronic signatures

- [ETSI] ETSI MCOMM Specialist Task Force 221
- [Federr2003] H. Fedderath: Digitale Signatur und Public Key Infrastruktur, <http://www-sec.uni-regensburg.de/security/5PKI.pdf>
- [FSEID2004] Project "Feasibility Study Electronic Identity Card", http://www.uni-kassel.de/fb10/oeff_recht/english/projekte/projekteDigiPerso_eng.gkhk
- [Fritsch2002] L. Fritsch.: A secure, economic infrastructure for signing of web based documents and financial affairs; CBL – Cyberbanking & Law, issue 2/2002;
- [FrRaRo2003] L. Fritsch, J. Ranke, and H. Rossnagel: Qualified Mobile Electronic Signatures: Possible, but worth a try? In: Information Security Solutions Europe (ISSE) 2003 Conference, Vienna Austria
- [FSMR2003] S. Figge, G. Schrott, J. Muntermann, and K. Rannenber: EARNING M-ONEY – A Situation based Approach for Mobile Business Models; In: Proceedings of the 11th European Conference on Information Systems (ECIS) 2003
- [FuFr2000] T. Fuchß, L. Fritsch: Security Certificates as a tool for reliably software engineering; Datenschutz und Datensicherheit 9/2000, pp.514ff.
- [GuD2004] Giesecke & Devrient: STARSIM® Applications, STARSIM®banking; www.gdm.de/eng/products/04/index.php4?product_id=386
- [GSM2004] GSM Association: GSM Statistics www.gsmworld.com/news/statistics/index.shtml
- [Perseu2004] B.Pfitzmann, C. Stübke: PERSEUS: A Quick Open-Source Path to Secure Electronic Signatures, <http://www.perseus-os.org/>
- [Pocket2004] Windows Mobile – based Pocket PCs, <http://www.microsoft.com/windowsmobile/products/pocketpc/default.mspx>
- [Raddic2004] Radicchio, <http://www.radicchio.org>
- [RaFrRo2003] J. Ranke, L. Fritsch, H. Rossnagel: M-Signaturen aus rechtlicher Sicht. In: Datenschutz und Datensicherheit 27 (2003) 2, p.95-100, Vieweg & Sohn
- [Rann2003] K. Rannenber: Identity Management in Mobile Applications In: Datenschutz und Datensicherheit 27 (2003) 9 (DuD), pp.546-550, Vieweg & Sohn
- [RegTP2004] Regulierungsbehörde für Telekommunikation und Post (RegTP) der Bundesrepublik Deutschland; <http://www.regtp.de/>
- [Symbian2004] Symbian OS – the mobile operating system, <http://www.symbian.com>
- [TMO2004] T-Mobile: Czech Republic: m-payment becomes a universal payment tool for customers; www.t-mobile.net/CDA/news_details,20,0,newsid-799,en.html?w=925&h=588
- [WAPF2004] WAP Forum: Specifications of WAP, WIM; <http://www.wapforum.org/>
- [WiTness] European IST Project „Wireless Trust for Europe“(WiTness), www.wireless-trust.org

Performance Evaluation of Certificate Based Authentication in Integrated Emerging 3G and Wi-Fi Networks

Georgios Kambourakis¹, Angelos Rouskas¹, and Dimitris Gritzalis²

¹Department of Information and Communication Systems Engineering,
University of the Aegean, Samos 83200, Greece
{gkamb, arouskas}@aegean.gr

²Department of Informatics,
Athens University of Economics and Business,
76 Patission St., Athens GR-10434, Greece
dgrit@aueb.gr

Abstract. Certificate based authentication of parties provides a powerful means for verifying claimed identities, avoiding the necessity of distributing shared secrets beforehand. Whereas Wi-Fi networks present security deficiencies, they manage to highly penetrate into the wireless market in a great degree due to their low cost, easy administration, great capacity, IP-oriented nature, etc. Considering Wi-Fi networking settings, administrated by different operators, as parts of a common core 3G infrastructure, the paper proposes and evaluates the potential application of enhanced TLS-based authentication mechanisms in integrated emerging-3G and Wi-Fi networks. We propose to use EAP-TLS protocol seconded by Public Key Infrastructure entities, to provide users with robust authentication mechanisms in hybrid WLAN-3G heterogeneous environment. Our alternative solution is discussed against EAP-AKA procedures as they appear in the latest 3G and integrated 3G/Wi-Fi specifications. Finally, the proposed mechanism is evaluated through a properly designed experimental test bed setup. *Keywords:* AKA; EAP; TLS; PKI; UMTS; WLAN.

1 Introduction

According to beyond-3G (B3G) vision, an IP backbone will constitute the core network for all heterogeneous wireless technologies and secure communication provision like confidentiality, access control and entity authentication, will become one of the major goals of these systems. Thus, certificate based authentication is attractive to support roaming in future mobile communications. Furthermore, the concept of digital signatures, that requires the usage of certificates, allows the introduction of complex many-to-many business models.

Recent works indicate that both a performance efficient implementation of TLS protocol for handheld devices is feasible [1] and secure, flexible and reconfigurable Authentication and Key Agreement (AKA) procedures for beyond 3G wireless communication networks can be implemented [2].

The paper discusses the application of TLS-based authentication into integrated 3G and Wi-Fi networks to provide strong end-to-end security and authentication to the user. The proposed application enables a Wi-Fi user, who is also a subscriber to a 3G mobile network operator, to move across Wi-Fi hot spots administrated by different WLAN operators. From a technical aspect, this application requires that all Wi-Fi networking settings are *loosely* or *tightly* incorporated [3] into a common core 3G infrastructure, while from a business point of view it is necessary that appropriate Roaming Agreements (RAs) are established among the various visited WLAN operators and the home 3G operator.

The rest of the paper is organised as follows: Taking into consideration Third Generation Partnership Project (3GPP)'s proposals and specifications for interworking and handover between IEEE 802.11a/b/c/i and 3G networks, the next Section discusses how a Wi-Fi networking setting can be integrated in a 3G infrastructure. Sections 3, 4 and 5 analyze and evaluate an AKA mechanism based on EAP-TLS, which can be applied into Wi-Fi settings and the paper is concluded in Section 6.

2 Wi-Fi Networks and 3G Infrastructures Integration

Emerging or B3G architectures are envisaged to constitute of an IP-based core network, whereas the access network can be based on a variety of heterogeneous wireless technologies depending on the nature of the access cell. The anticipated provision of many *uncoordinated* Wi-Fi picocells where coverage is limited e.g. within buildings, will bring to foreground many open issues concerning authentication and security, mobility management, roaming and billing of mobile users moving among different Wi-Fi settings.

Our work deals with *authentication* issues in different Wi-Fi operators and proposes the application of B3G TLS based authentication mechanisms into Wi-Fi networks. We suppose that the Authentication, Authorization and Accounting (AAA) procedures of a mobile Wi-Fi user can be controlled in a “centralized” or “semi-centralized” way by his Home core 3G network. A WLAN user needs to know only his home 3G network operator, who is responsible to establish and maintain RAs with various ending WLAN operators. Depending on the RA between the two operators, the user may receive Internet access through his home 3G network (via the Wi-Fi network) or directly through the current Wi-Fi access network, after being authenticated by his Home 3G network. Obviously, such a solution also assumes that the user has a dual mode mobile station supporting both WLAN and UMTS, or the WLAN device can be linked with a UE, which supports USIM capabilities (Bluetooth, USB, IrDa).

Current 3GPP specifications for Universal Mobile Telecommunications System (UMTS) Release 6[4], describe an interworking architecture (Figure 1) where the home network is responsible for access control, while 3GPP proxy AAA relays access control signalling to the Home 3GPP AAA server. UMTS-SIM (USIM) based authentication mechanism can be based on the existing UMTS AKA method. As this method should be independent of the underlying WLAN standard and should be supported by a standard authentication mechanism, 3GPP seems to choose the EAP-AKA protocol described in [4, 5]. EAP is a general protocol for PPP authentication, which can sup-

port multiple authentication mechanisms. Consequently, EAP-AKA provides a way to exchange AKA authentication messages encapsulated within the EAP protocol.

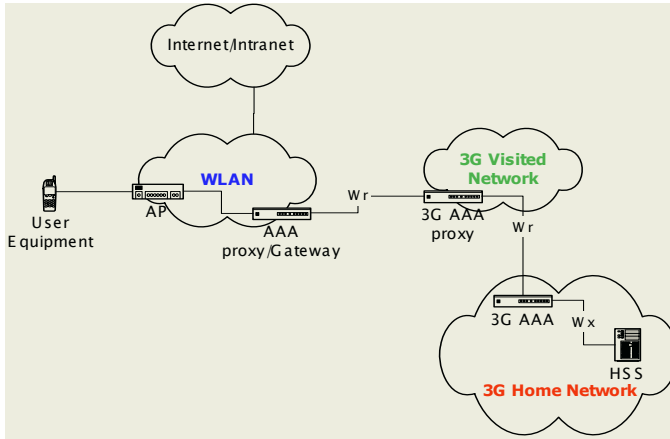


Fig. 1. Wi-Fi integration in UMTS concept

In case the Serving Network (SN) is a WLAN, the mobile terminal is connected to an Access Point (AP). The user presents his Network Access Identifier (NAI), which is of the form `IMSI@domain` or `Packet_TMSI@domain`. The access request is forwarded to the AAA proxy that translates the AAA request into the equivalent 3G AAA protocol request. Note that this Proxy or Gateway might be pre-configured or dynamically searched. The procedure may cross several other authentication domains. Usually the EAP server is separate from the authenticator node, which resides closest to the user's machine (also called *Supplicant*) e.g. an AP or an 802.1X bridge. The supplicant communicates with the AAA server that provides EAP server functionality using an AAA protocol, such as RADIUS or DIAMETER.

This approach has the main advantage that mobility management, roaming, billing and location issues are under the supervision of the “master” UMTS network. An enhanced system would also require support for “vertical” handover between WLAN and UMTS. This approach also minimizes the necessary changes to the existing 3G network core elements (e.g. HSS, GGSN).

From the user's security standpoint, USIM based authentication of a subscriber for WLAN services, offers two significant benefits: (a) easy integration of the WLAN's subscriber credentials, to 3G Home Subscriber Server (HSS), as those are of identical format to 2.5/3G, and (b) WLAN's security level equal to that offered by 2.5/3G, thereby resolving the drawbacks of current IEEE 802.11 protocols [6, 7].

However, as it is desirable to support mutual authentication and since EAP-AKA assumes the existence of a long-term symmetric key per subscriber, it is useful to have a mechanism for session key establishment. Introducing TLS, we can take advantage of the protected and flexible ciphersuite negotiation, mutual certificate based authentication and scalable key management capabilities, in order to provide strong authentication and end-to-end security to the user of this heterogeneous architecture.

Moreover, although several known weaknesses in GSM AKA seem to be now fixed in UMTS, there are still some inefficiencies which affect the EAP-AKA authentication mechanism too. For a detailed breakdown of 3G-AKA and EAP-AKA shortcomings refer to [2, 9, 10, 11] & [5] respectively.

3 EAP-TLS and PKI in Heterogeneous Mobile Environments

EAP-TLS [12] is based on SSL Version 3.0, and the SSL handshake is performed over EAP, whereas on the Internet the handshake is carried out over TCP. As EAP-TLS performs mutual SSL authentication, each side is required to prove its identity to the other using its certificate and its private key.

Certainly to implement an AKA mechanism based on TLS, we need some sort of public key infrastructure (PKI), which is not necessarily part of the 3G network core. Integration between 3G mobile systems and PKI has not been standardized yet, but recent 3GPP discussion documents [13] deal with that particular subject. Successful wireless PKI implementations and solutions from companies like Sonera Smarttrust, Lucent Technologies and Entrust, strengthen the assertion that PKI has become an acknowledged and promising component of standards. Projects like ASPECT [14] and USECA [15], 3GPP discussion papers especially for UMTS R6, as well as other papers [16], foresee that evolution. The eNorge 2005 strategy calls for a shared PKI for Norway, while advanced standards such MexE, WAP and i-mode from NTT DoCoMo have moved forward to introduce public key methods. More on PKI and 3G integration requirements can be found in [2, 13, 16, 17, 18].

Performance considerations have held from using TLS in resource-constrained environments, like the wireless one. Nevertheless, the necessity for more processing power and memory, has driven smart cards toward more advanced architectures, all the way to where we are beginning to see 32-bit RISC-based ARM processors in smart cards. These cards can effectively store and protect the subscriber's private key, generate good pseudo-random values and take over of symmetric key (un)+wrapping functions. Mobile's device processor can efficiently carry out the rest of the calculations needed by TLS protocol. A recent study has also shown the feasibility of TLS in handheld wireless devices [1], while relevant work showed that SSL's handshake protocol time can be improved up to 5.7X times [19].

TLS supports different protocols for creating pre-master keys (RSA, Diffie-Hellman, etc), several different cryptographic algorithms and two different MAC algorithms. In the context of an AKA procedure, these properties can provide the appropriate flexibility in an integrated 3G-WLAN environment, when the available means (from a perspective of diversity and computational power) at the attackers side are increasing rapidly.

4 An AKA Mechanism Based on EAP-TLS

Motivated by the aforementioned technological trends, we propose an alternative AKA procedure based on EAP-TLS, instead of EAP-AKA, for integrated 3G/Wi-Fi

networks. Figure 2, depicts the exchange of protocol messages, including the essential adaptations to make it ‘mobile-enabled’ and focusing on public key operations in the supplicant side which is generally considered as computational weak.

The appropriate 3G AAA server is chosen based on the NAI. Note, that since the client claimed his identity in the EAP-Response Identity packet, the EAP server should verify that the claimed identity corresponds to the certificate presented by the peer. This means that user ID must be included in the peer certificate. From the AAA server side, a mapping from the temporary identifier (P-TMSI) to the IMSI is required too. Likewise, supplicant must check against EAP’s server certificate validity (Expiration time / Name / Signed by trusted CA etc). For a detailed pure EAP-TLS protocol explanation, refer to [12].

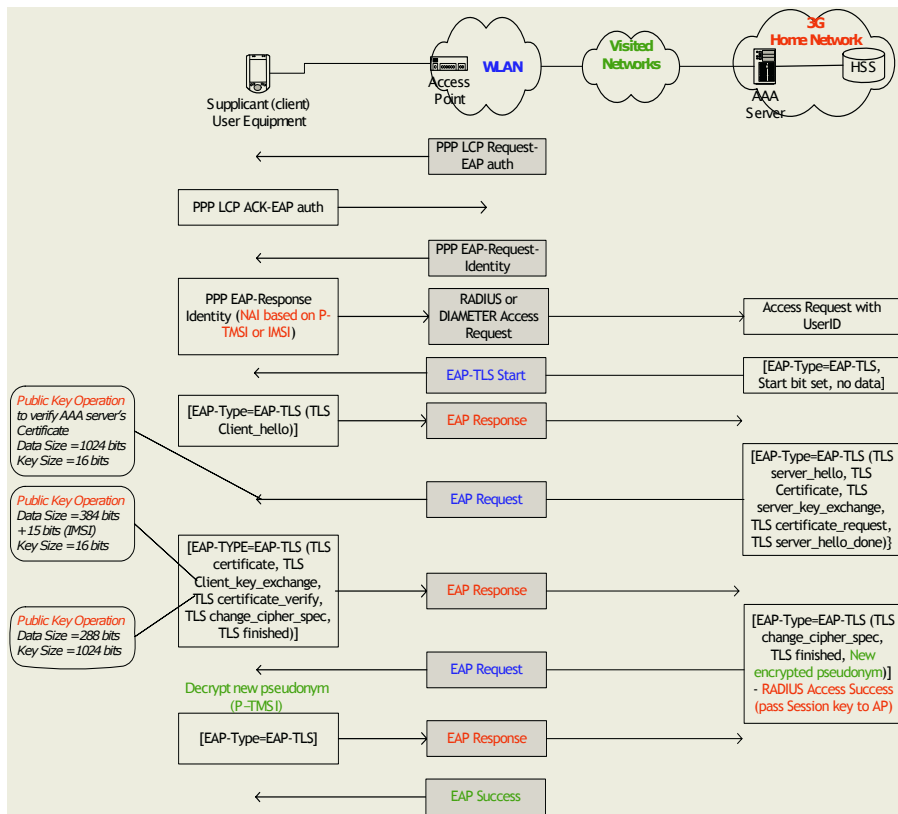


Fig. 2. AKA Mechanism based on EAP-TLS

When comparing the two available options, EAP-AKA and EAP-TLS, we can highlight the following observations:

- The 3GPP network architecture to support integration remains the same with the addition of the underlying PKI. As shown in Figure 3, a CA can be connected with the 3G-core network, either through GGSN (“natural” option), SGSN, Proxy or

Serving Call State Control Function (P-CSCF / S-CSCF) or with the addition of a new gateway element, which is connected to AAA server. This last option guarantees minimal changes to 3G-core network elements. In any case, new IP interfaces have to be created accordingly.

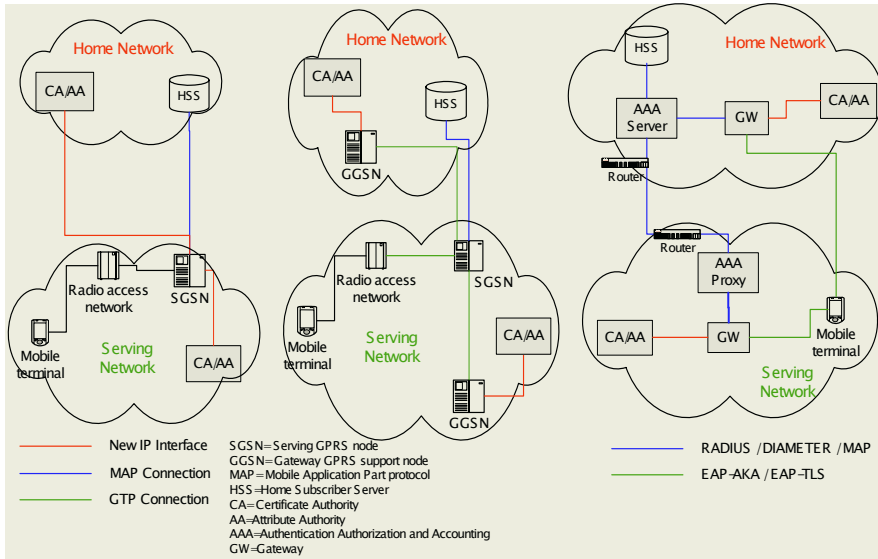


Fig. 3. 3G Architectures to support PKI

- The supplicant and the AAA server must support EAP-TLS, while the AP has to support EAP-TLS authentication. Currently, EAP-TLS protocol is becoming widely supported by the most accredited vendors in routers, APs and end user terminals, ensuring minimal changes and easy integration.
- Any AAA server (WLAN or 3G) that resides near the supplicant can provide for authentication, thus improving mobility. This is possible as the “Any-AAA server” can exchange (offline) cross-reference certificates with the home AAA server, or both can have a signed certificate from a common (Root) CA. Accounting details could be “batch transferred”, according to bilateral pre-arrangements.
- Supplicant certificate revocation can be handled by IMSI, thus avoiding CRLs and related procedures.
- The overall performance can be significantly enhanced, using TLS option for session resumption. The purpose of sessionID included in supplicant hello, is to allow for improved efficiency in the case where a client repeatedly attempts to authenticate to an EAP server within a short period of time. Based on the sessionID chosen by the peer, and the time elapsed since the previous authentication, the EAP server will decide whether the proposed session should be resumed or not. Recent studies also showed that session reuse could be further improved, using a TLS session aware dispatcher, when the operator is planning to install a cluster of TLS authentication servers [20].

- TLS protocol has proved its effectiveness in the wired Internet, and, seconded by PKI, is best suited to support large heterogeneous infrastructures. The flexibility to choose among several ciphersuites and built in MAC algorithms decrease the possibility of intrusions. For instance, using ephemeral Diffie-Hellman key exchange can support forward secrecy. Furthermore, the scalability of public key mechanisms offers a competitive framework to overcome symmetric key based security inefficiencies. Last but not least, PKI add-on value services, like the use of Attribute Certificates (AC) are also possible [21].
- There is no need for HSS to generate and distribute authentication quintuplets, thus avoiding the risk to be stolen or spoiled. On the other hand, certificates control mutual authentication process.
- AKA-TLS has to be generally considered as an end-to-end authentication procedure in contrast to EAP-AKA, which provides a *hop-by-hop* fashioned security, as intermediate devices should implement IPsec, MAPsec or SSL to secure inter or intra network communications.

5 EAP-TLS Service Time Evaluation

5.1 Test Bed Setup

In order to evaluate the performance of EAP-TLS AKA mechanism in terms of service times, we constructed experimental hardware architecture. The development and test model topology is illustrated in Figure 4. The presumed mobile device is a low-end laptop machine that uses Windows XP Home edition operating system. The supplicant incorporates a Pentium II 400MHz CPU and has 80 MB of RAM available. At the other end, the RADIUS server machine has a Pentium 4 1.4 MHz processor and 128 MB RAM, running Linux Slackware 9.1 operating system. To implement RADIUS server functionality we used the well known open-source package Freeradius in version 0.9.3.

The client is connected to the network using an 802.11g wireless PCMCIA card and an 802.11g AP. The RADIUS server and the AP reside to different networks, acting as the Visited (foreign) and Home networks for the client. The average ping time between the two networks, measured with a ping tool was about 120 msec. Another laptop machine which is connected to the same network with the RADIUS server, generates a large number of EAP-TLS requests to virtually load the server. The times between successive EAP-TLS authentication requests follow the negative exponential distribution. Finally, the necessary for our experiments certificates, was constructed using the well known Apache style license OpenSSL toolkit in version 0.9.7c.

The handshake and authentication procedures are mutual, meaning that both the supplicant and the RADIUS server exchange their certificates, which are kept locally along with the corresponding trusted CAs public keys list. We used a depth-two certificate chain schema in order to weigh up a serving or visited network authentication schema. Both parties check certificates validity, against time expiration and issuing CA (trust anchor). Neither party check certificates validity against any revocation list.

Only the RADIUS server is bound to do so by checking against supplicant's P-TMSI (mapping it to the correct IMSI).

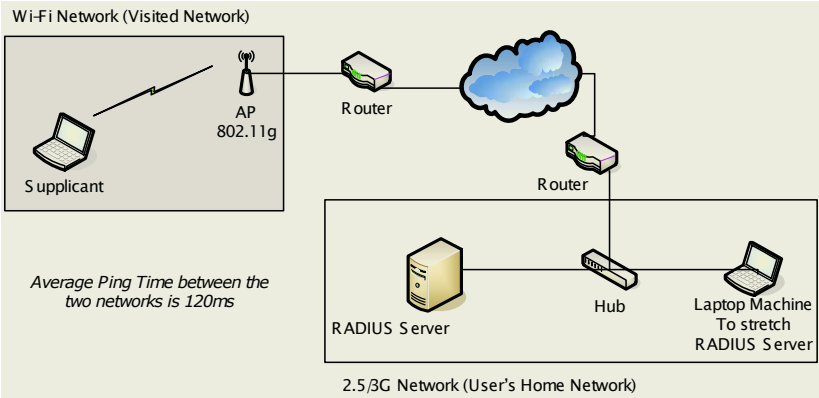


Fig. 4. Test bed Topology

5.2 Measurements Results

We run our experiments with various values of the request arrival rate λ (transactions per second) for the process which adds virtual load to the server. We gathered 1000 measurements from an equal number of transactions initiated by the supplicant, using the popular network analyzer Ethereal in version 0.10.0 and RADIUS server log files. The average initialization time for the supplicant was 0.062 sec. We present the corresponding average total EAP-TLS authentication service time values in seconds for different scenarios we tested, in Table 1 and Figure 5.

Table 1. Average Total EAP-TLS Service Times in Seconds

Virtual Load (λ)	AP in the same room with supplicant	Supplicant outside the building
1	4.468	4.575
4	4.524	4.604
5	4.694	4.625
8	4.737	4.695
10	4.897	4.753
20	4.917	4.959
25	5.036	5.045
50	5.761	5.912
Average Time	4.879	4.896

An average time below 5 sec, as it appears in Table 1, is certainly an acceptable authentication time duration for the users of a mobile B3G device, since a real 2.5G standard AKA mechanism, assuming that someone activates his device in a roaming, network, takes about 4 – 7 sec to complete. Nevertheless, EAP-TLS authentication

time is expected to grow according to the Home and Serving networks 'distance'. The greater the distance measured in ping times is, the greater the EAP-TLS authentication time is expected to be. This is mainly due to TLS protocol round-trips. Moreover, during the TLS handshake the server must wait for a client message and vice versa. Thus, it is often computationally cheaper and network faster (considering round-trip times) to generate a number of TLS messages and transmit them all at once [22].

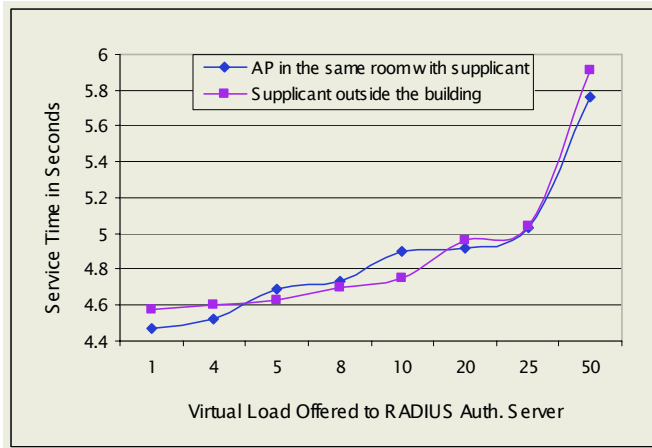


Fig. 5. EAP-TLS Service Times

It is also obvious, that the distance between the AP and the supplicant has a negligible effect on EAP-TLS authentication time. We discovered that this is true as long as the signal quality remain above 55 – 60%. Furthermore, the size of the certificates and its depth, not only decelerates the certificate verification process in each party, but also at the same time adds several extra bytes to the relevant handshake messages. Finally, an extra delay is expected if the RADIUS server is located remotely to user's Home network HSS. In that case, the RADIUS server has to communicate with the subsequent HSS in order to fetch user's credentials and successfully authenticate the user.

6 Conclusions

With the ongoing development in the area of mobile communication technologies we are approaching the 'all-IP' 4G vision step by step. In this paper we considered the authentication problem faced by 3G mobile roaming subscribers who need to access wireless Internet services through Wi-Fi hot spots administrated by different operators. We argued on the application of EAP-TLS in contrast to EAP-AKA based authentication mechanisms into integrated 3G and Wi-Fi networks. Through experimentation we discovered that EAP-TLS authentication is attainable in terms of service times. Our proposed solution overcomes 3G and WLAN authentication inefficiencies, while users are also offered the possibility to enjoy add-on value services, which stem

from PKI incorporation. Topics to be further investigated include evaluation scenarios with Pocket PC machines, certificate-depths above two and RADIUS – HSS communication.

Acknowledgments

We would like to thank Mr. Lizos Konstantinos for providing us with the network measurements.

7 References

1. Gupta V. & Gupta S., Experiments in Wireless Internet Security, *In the Proc. Of IEEE Wireless Communications and Networking Conf. (WCNC 2002)*, no. 1, pp. 859-863, March 2002.
2. Kambourakis G., Rouskas A., & Gritzalis S., "Using SSL in Authentication and Key Agreement Procedures of Future Mobile Networks", *In the Proc. of the 4th IEEE Int'l Conf. On Mobile and Wireless Comm. Networks (MWCN 2002)*, pp. 152-156, Sep. 2002.
3. Salkintzis, A., Fors, C. & Pazhyannur, R., "WLAN-GPRS Integration for Next-Generation Mobile Data Networks", *IEEE Wireless Communications Magazine*, pp. 112-124, Oct 2002.
4. 3GPP Technical Specification, WLAN Interworking Security, (TS 33.cde v0.1.0), July 2002.
5. Arkko, J. and Haverinen, H., "EAP-AKA Authentication", <draft-arkko-pppext-eap-aka - 11.txt>, Oct. 2003.
6. M. Gast, "802.11 Wireless Networks: The Definitive Guide", O'Reilly, April 2002.
7. D. Eaton, "Diving into the 802.11i Spec: A Tutorial". Feb 2003, electronically available in http://www.commsdesign.com/design_corner/OEG20021126S0003.
8. 3GPP Technical Specs, A guide to 3rd Generation Security, (TR 33.900 v.1.2.0), Jan. 2000.
9. Aamodt, T., Friiso, T., Koien, G., "Security in UMTS-Integrity", Telenor R&D, Feb. 2001.
10. Niemi, V. & Nyberg, K., *UMTS Security*, Wiley, 2003.
11. 3GPP Technical Specs, 3G Security Architecture, TS 33.102 v.5.1.0), December 2002.
12. IETF RFC 2716, "PPP EAP-TLS Authentication Protocol", Oct. 1999.
13. 3GPP TSG, "Architecture proposal to support subscriber certificates", Discussion and Approval document, Tdoc S2-022854, Oct. 2002.
14. ASPECT Project, Securing the future of mobile communications, www.esat.kuleuven.ac.be/cosic/aspect, 1999.
15. USECA Project, UMTS Security Architecture: Intermediate report on a PKI architecture for UMTS, Public Report, July 1999.
16. Kambourakis G., Rouskas A., Gritzalis S., "Introducing PKI to enhance Security in Future Mobile Networks", *in the Proc. of the IFIPSEC'2003 18th IFIP Int'l Information Security Conf.*, pp.109-120, Athens, Greece May 2003.
17. 3GPP TSG, "Using PKI to provide network domain security", Discussion Document (S3-010622 SA WG3 Security – S3#15bis), Nov. 2000.
18. 3GPP Technical Specs, Bootstrapping of application security using AKA and Support of Subscriber Certificates: System Description, (TS ab.cde v.3.0), Sep. 2003.
19. Nachiketh, P., Srivaths, R., Anand, R. & Ganesh, L., "Optimizing Public-Key Encryption for Wireless Clients", *In the Proc. of the IEEE Int'l Conf. On Communications (ICC 2002)*, no 1, pp. 1050 – 1056, April 2002.
20. Apostolopoulos, G. et al., Securing Electronic Commerce: Reducing the SSL Overhead, *IEEE Network Magazine*, no 4, pp. 8-16, July/August 2000.
21. 3GPP TSG, "Support of certificates in 3GPP security Architecture", Discussion Document S3-010353 SA WG3 Security – S3#19, July 2001.
22. Rescorla, E., *SSL and TLS Designing and Building Secure Systems*, Addison-Wesley, 2001

A Credential Conversion Service for SAML-based Scenarios^{*}

Óscar Cánovas¹, Gabriel López², and Antonio F. Gómez-Skarmeta²

¹ Department of Computer Engineering
ocanovas@dittec.um.es

² Department of Information and Communications Engineering
University of Murcia, Spain
{gabilm, skarmeta}@dif.um.es

Abstract. Coordination of different administrative domains involves several security concerns, especially from an authorization point of view. SAML is getting a lot of popularity as a language that can be used to bridge several isolated authorization systems in order to provide a common interface in a shared target scenario. In this paper, we present a Credential Conversion Service (CCS) that converts non-SAML credentials into SAML assertions following the rules of a conversion policy. CCS provides two different profiles governing how to exchange SAML assertions, and also defines some extensions to SAML in order to express the syntax and semantics of our CCS.

1 Introduction and Rationale

There are a number of authorization proposals for use on several areas of computing, such as X.509 Attribute Certificates [5], SPKI/SDSI certificates [4], or SAML assertions [8]. Based on those proposals, we can find in the literature several real systems providing authorization services for different scenarios, such as PERMIS [2], DCMS [3], or CAS [10]. From an abstract point of view, all of them have similar semantics, since they are based on given descriptions of subjects, actions being requested, information about the target resource, contextual data, delegation conditions, etc. However, there are also several differences that make them incompatible, such as supporting protocols, specification languages, encoding formats, authorization semantics, etc.

On the other hand, during the last few years, we have experienced an increasing emergence of XML-based technologies in research areas related to security. SAML (Security Assertion Markup Language) [8] is an XML-based standard for requesting and expressing authorization assertions that is getting more and more acceptance in several fields, such as the Open Grid Services Architecture (OGSA) [12] or some architectures for access control to web resources [1].

As we show in this paper, we have defined a Credential Conversion Service (CCS) that integrates external authorization schemes (non SAML-based) into

^{*} Partially supported by IST-2001-32161, IST-2002-001929 and PB/32/FS/02

authorization scenarios which make use of SAML as the main language for assertions. Our starting point is an end user requesting access to a resource secured in a SAML environment. We do not want the user authorities pertaining to non-SAML domains to issue SAML assertions, since they were not designed to perform that task. In fact, what we need is a conversion service able to translate the external credentials into SAML assertions. CCS defines the different entities involved in that process, their relationships, and two different profiles based on push and pull models. CCS extends some standard SAML elements, such as assertions and queries, to provide the needed syntax and semantics.

The conversion process can generate different types of assertions, depending on the credentials provided by the user. Identity certificates can be converted into Authentication Statements, attribute or role membership certificates might be translated into Attribute Statements, and authorization certificates could generate Authorization Decision Statements. Those resulting assertions can be stored in different locations, ranging from the user authority, the user himself or a repository belonging to the target scenario. As we present in next sections, our CCS profiles and statements are flexible enough to reflect all those possibilities.

2 Context and Architectural Entities

In our context, there is a SAML-based administrative domain, named *target scenario*, which contains a set of resources that must be made available to the end users of another different administrative domain, named *source domain*, based on a non-SAML authorization mechanism. This configuration can be observed in Virtual Organizations (VO) of Grid Computing environments, where the authorization mechanisms protecting the shared resources are being standardized using SAML, in spite of the fact that some of the participating organizations have already deployed authorization systems based on X.509 ACs or SPKI/SDSI certificates. In order to link these two different environments, we can consider several alternatives from an authorization management point of view:

1. Policy enforcement points (PEP) and policy decision points (PDP) of the target scenario must be able to understand non-SAML credentials. In this way, no conversion is needed and the end users of the source domain can use their existing privileges. This option presents several problems derived from scalability and complexity (bridging policies must be made widely available).
2. Authorities of the source domain must be able to generate SAML assertions containing the privileges related to their end users, which might be further interpreted by the PEPs and PDPs of the target scenario. However, this solution presents two main drawbacks. First, every authority in the source domain must be modified in order to support SAML. Second, this involves that these authorities should know lots of details about the target scenario.
3. End users make use of a special credential conversion service, provided by the target scenario, in order to adapt their source credentials to SAML assertions. In this way, the conversion (or bridging) policy is enforced by a central

element (although the CCS can be replicated among different nodes), minimizing the drawbacks associated to scalability and complexity. CCS must be trusted by the PEPs and PDPs of the target scenario.

Communications between the source domain and the CCS have been designed taking into account the degree of adoption of the existing proposals for authorization. Nowadays, Web browsers and other common networking applications do not provide any kind of support for authorization certificates. The resulting systems are, therefore, mainly based on the pull operation mode, since users cannot push their credentials, and these statements must be pulled from repositories or authorities. Consequently, our CCS will assume that end users are not able to provide their credentials by themselves.

All these requirements were considered during the design stage of the CCS architecture, which is composed by three architectural entities involved in the conversion process:

- *End users* use a standard Web browser and authenticate to an authority of the source domain (named *user authority*) by some means outside the scope of SAML (in our case, by means of a fully authenticated connection based on HTTPS). Once they have been authenticated, they send to the user authority a message containing a description about the target scenario and the resources to be accessed.
- *User Authorities* are responsible for determining the credentials related to the target scenario, and must create the conversion query to be sent to the CCS. Following the same scheme presented in the standard profiles of SAML [9], conversion queries are not directly transmitted to the CCS, but through the user.
- The *CCS* processes the conversion queries according to the conversion policy specified in the target scenario. The resulting assertions can be transmitted to different receivers, such as end users, user authorities or entities pertaining to the target scenario (repositories or PDPs).

Once the credentials have been converted, the standard SAML profiles can be used to transmit the resulting assertions to the destination sites.

3 Profiles for CCS

Profiles [9] are sets of rules describing how to embed SAML assertions into and extract them from a framework or protocol. Our profiles for CCS describes how the SAML assertions and the external credentials are combined, and also the way they are exchanged between the participating entities. This paper defines two different profiles for CCS, named *Push Conversion* and *Pull Conversion*.

In both profiles, the end user accesses the user authority by means of a fully-authenticated HTTP/TLS link. During this initial step, the user must specify the target scenario or a set of credentials to be converted. As we commented previously, the user authority should be equipped with a communication module

able to generate SAML protocol messages, and should be configured with the location, i.e. a URL, of the CCS related to the target scenario. Connections between end users and the CCS must be over HTTP/TLS, and connections between the CCS and the authorities should be over SOAP/TLS.

3.1 Push Conversion

During the push conversion, the credentials are pushed from the user authority to the CCS through the user, as we can see in Figure 1. After the initial step, once the HTTPS connection has been established and parties are authenticated, the user authority gets the reference to the target scenario or to the related credentials. These credentials are wrapped into a *ccs:WrappedStatement*, which is an extension of an *saml:Statement* designed to encapsulate non-SAML credentials (details about the extensions will be provided in Section 4).

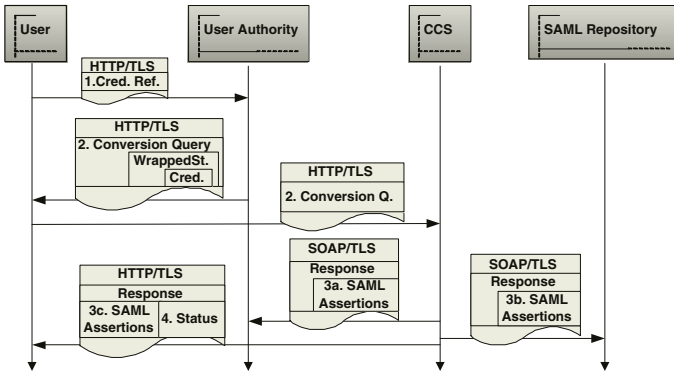


Fig. 1. Push Conversion

Communications between the user authority and the CCS must be performed using the SAML protocols based on queries and responses. Therefore, in order to transport the wrapped statement, the authority has to use the *samlp:Request* message. We have extended the *saml:SubjectQuery* to create a specific *ccs:ConversionQuery*, which is designed to fulfill two main requirements: on the one hand, it has to provide support not only to the push conversion, i.e. containing wrapped statements, but also to the pull conversion, i.e. transporting artifacts; on the other hand, it specifies the intended recipient of the assertions converted by the CCS (end users, user authorities, or SAML repositories). The request containing the conversion query must also define which kind of assertion is expected to be generated by the CCS (authentication, attribute, or authorization decision statements).

Thus, the second step includes a HTTP response from the user authority containing a HTML form with the SAML request and a pointer to the CCS.

The end user has to submit the form using a HTTP POST message. Once the CCS has obtained the query, it can perform the conversion to the corresponding assertions, if success, or it can generate a response indicating that the request cannot fulfill the requirements imposed by the conversion policy.

When a *saml:AuthorizationDecisionStatement* is generated as response, the assertion includes the target resource identifier, defines the action being granted, makes reference to the subject, and might optionally contain, as evidence, a reference to the wrapped statements used to derive the new assertions. On the other hand, if the assertions generated by CCS are *saml:AttributeStatements*, they must contain the attributes conforming with the conversion policy, translated into the corresponding attribute designators of the target scenario. Finally, conversion of identity certificates involves the creation of *saml:AuthenticationStatements* which may include different authentication methods and identifiers depending on the source public key certificates.

As we can see in Figure 1, once the assertions have been generated, they can be transmitted to three different receivers. The line labeled as *3a* makes reference to those situations where the user authority can also store SAML assertions that will be later exchanged by means of one of the standard SAML profiles (post or artifact). Otherwise, the line tagged as *3b* represents the most common option, that is, the use of a repository belonging to the target scenario. Finally, assertions might be transmitted to the end users (line *3c*). Regardless of the specific receiver, in step 4 the end user receives a status message (a SAML response) announcing whether the conversion has been processed successfully.

3.2 Pull Conversion

In the pull model, the user authority includes in the conversion query a link or reference to those credentials related to the target scenario (following the SAML notation, we will use the term *artifact*). The artifact format, specified in [9], contains a 20-byte sequence used to determine the user authority and its location, and a 20-byte sequence that makes reference to the credential to be converted. Conversion queries can include several artifacts to make reference to several credentials that might be useful in the target scenario.

As we can see in Figure 2, once the CCS has obtained the conversion query, it uses a standard *samlp:Request* containing the artifact in order to obtain from the user authority the related credential. The user authority must generate a *samlp:Response* including the credential as a *WrappedStatement* (if an error occurred, the response contains no wrapped statements and an appropriate status code). If the conversion query specifies more than one artifact, steps 3 and 4 are repeated as many times as needed. Once the CCS has obtained the wrapped statements, the profile continues like the push conversion.

The pull profile is especially useful in systems that hide the details about the existing credentials from the users. In this way, credentials containing confidential information about privileges are always handled by internal and trusted entities.

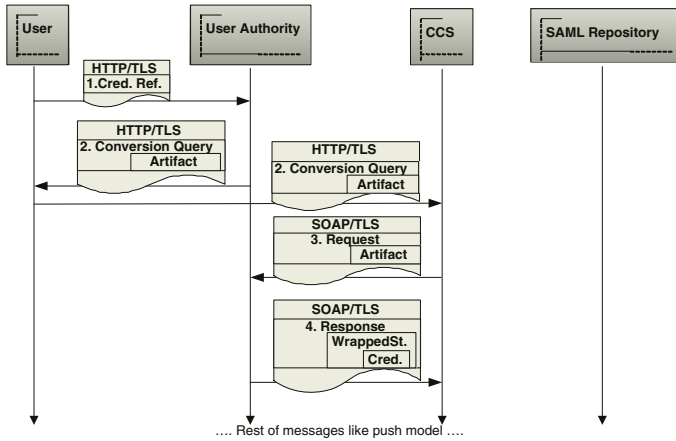


Fig. 2. Pull Conversion

4 SAML Extensions for CCS

Our CCS is based on operations that cannot be described using the standard SAML specification. Therefore, we decided to incorporate the functionality related to CCS by means of some extensions to existing SAML elements. We can summarize our extensions in two new SAML elements defined in the CCS namespace: *ccs:WrappedStatement* and *ccs:ConversionQuery*.

4.1 Encapsulation of Credentials: *WrappedStatement*

As we have shown, the user authority has to send the non-SAML credentials, or a reference to them, to the CCS in order to start the conversion process. We use a mechanism to wrap the credentials into new SAML statements named *WrappedStatements*. This is the defined schema:

```

<element name="WrappedStatement" type="ccs:WrappedStatementType"/>
<complexType name="WrappedStatementType">
  <complexContent>
    <extension base="saml:StatementAbstractType">
      <sequence>
        <element name="WrappedData" type="hexBinary" minOccurs="1"
          maxOccurs="unbounded"/>
      </sequence>
      <attribute name="StatementType" type="anyURI" use="required"/>
      <attribute name="Encoding" type="anyURI" use="required"/>
    </extension>
  </complexContent>
</complexType>

```

- *StatementType* defines the type of credential being wrapped. Following the SAML rules, it is defined as a URI (specific URIs for this field have the prefix `urn:ccs:names:credential`, and some defined values are `x509`, `x509ac`, `keynote`, `saml`, `sdsi`, `spki`).
- *Encoding* specifies the encoding used to express the credential. Usually, every certification standard has its own encoding formats, such as DER encoding for X.509 certificates, canonical s-expressions for SPKI certificates, or XML for SAML assertions. It is also defined as a URI.
- *WrappedData*. This field contains one or more credentials of type *StatementType*, which are formatted using *Encoding* method.

4.2 ConversionQuery

If we consider the standard queries defined in the specification for the SAML protocols, we can infer that none of them fulfills the requirements imposed by our CCS. *AuthenticationQuery* elements are mainly related to authentication validations, and include information about authentication methods, which is not necessary for our purposes. On the other hand, the elements for *AttributeQuery* and *AuthorizationDecisionQuery* include information about the resource, and, in our case, the resource has a different meaning since it represents the non-SAML credentials to be converted. Consequently, we decided to define a new type of query, named *ConversionQuery*, which is specified by the next schema:

```
<element name="ConversionQuery" type="ccs:ConversionQueryType"/>
<complexType name="ConversionQueryType">
  <complexContent>
    <extension base="samlp:SubjectQueryAbstractType">
      <choice>
        <element ref="samlp:AssertionArtifact" maxOccurs="unbounded"
          minOccurs="1"/>
        <element ref="saml:Assertion" maxOccurs="unbounded"
          minOccurs="1"/>
      </choice>
      <attribute name="Recipient" type="anyURI" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

In order to include information about the entity requesting the conversion of credentials, we decided to extend *saml:SubjectQuery*. Our *ConversionQuery* also includes the following components:

- *Recipient* is a new attribute defining the entity that will receive the resulting SAML assertions. Following the SAML rules, it is defined as a URI that makes reference to some well known architectural entity.

- The *samlp:RespondWith* element contained in every SAML request should be used to specify the type of SAML assertion that will be generated.
- In order to support conversions based on the push profile, the *ConversionQuery* includes an element to represent the *ccs:WrappedStatement*.
- If the pull profile is selected, the credentials are referenced by means of an *samlp:AssertionArtifact*.

It is worth noting that the request containing the *ConversionQuery* is digitally signed by the user authority in order to provide integrity.

5 Related Work

Despite its importance, relatively little systems-oriented research has addressed the issues related to credential conversion services. Several exceptions have informed our work. In [13] we can find a simplified example of the GT3 OGSA security model in action. In an OGSA scenario, when the end user determines that the needed credentials are not already present, he should contact a credential conversion service to convert existing credentials to the needed format, mechanism, and/or trust root. The proposed examples of such service are CAS [10], based on X.509 proxy certificates, for translating the user's personal credentials to VO credentials, and a mechanism for converting between Kerberos and PKI statements [7]. However, as described in the OGSA Security Roadmap [11], SAML is going to play a major role in the authorization services related to OGSA. Consequently, our CCS was based on conversion of public key credentials in pure SAML assertions, thus complementing the above mentioned proposals. Finally, it is worth noting that there are similar approaches to our CCS for Kerberos scenarios [6].

References

- [1] S. Cantor and M. Erdos. *Shibboleth-Architecture*, May 2002. Internet 2 Draft.
- [2] D. W. Chadwick, A. Otenko, and E. Ball. Role-Based access control with X.509 Attribute Certificates. *IEEE Internet Computing*, 7(2):62–69, March 2003.
- [3] O. Cnovas and A. F. Gmez. A Distributed Credential Management System for SPKI-Based Delegation Systems. In *Proceedings of 1st Annual PKI Research Workshop*, pages 65–76, April 2002.
- [4] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI certificate theory*, September 1999. Request For Comments (RFC) 2693.
- [5] S. Farrel and R. Housley. *An Internet Attribute Certificate Profile for Authorization*. Internet Engineering Task Force, April 2002. Request for Comments (RFC) 3281.
- [6] J. Hughes. *SAML 2.0 - Kerberos use cases*, September 2003. OASIS draft.
- [7] O. Kornievskiaia, P. Honeyman, B. Doster, and K. Coffman. Kerberized Credential Translation: A Solution to Web Access Control. In *Proceedings of 10th Usenix Security Symposium*, 2001.

- [8] E. Maler, P. Mishra, and R. Philpott. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, September 2003. OASIS Standard.
- [9] E. Maler, P. Mishra, and R. Philpott. *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1*, September 2003. OASIS Standard.
- [10] L. Pearlman, C. Kesselman, V. Welch, I. Foster, and S. Tuecke. The community authorization service: Status and future. In *Proceedings of CHEP03*, 2003.
- [11] F. Siebenlist, V. Welch, S. Tuecke, I. Foster, N. Nagarathnam, P. Janson, J. Dayka, and A. Nadalin. *OGSA Security Roadmap*. Internet Engineering Task Force, July 2002. Global Grid Forum Specification Roadmap towards a Secure OGSA.
- [12] V. Welch, F. Siebenlist, D. Chadwick, S. Meder, and L. Pearlman. *Use of SAML for OGSA Authorization*, January 2004. Global Grid Forum draft.
- [13] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, and K. Czajkowski. Security for grid services. In *Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing*, pages 48–57, 2003.

A New Design of Privilege Management Infrastructure with Binding Signature Semantics

Kemal Bicakci and Nazife Baykal

Middle East Technical University, Informatics Institute
06531 Ankara, Turkey
{bicakci,baykal}@ii.metu.edu.tr

Abstract. Just like PKI, used to support public key certificates, Privilege Management Infrastructure (PMI) is built to provide a foundation to employ attribute certificates. Although most of the PKI ideas can be applied to PMI as well, PMI has some unique characteristics for instance it should handle attributes containing confidential information. Motivating by this fact, Dawson et al. recently proposed a new PMI design for those who would like to use the outsourced PKI but keep the PMI management inside the organization. In this paper, we propose an alternative design to have a more fine-grained control over attribute certificates. Immediate revocation and simplified verification are two big advantages of our approach.

Keywords: digital signature, PMI, attribute certificate, revocation, binding signature semantics.

1 Introduction

Broadly speaking, one of the building blocks of modern cryptography, digital signatures can be used for two separate reasons. First, the conventional usage is for message authentication. Addressing also the non-repudiation problem, digital signatures are widely used for this purpose. Secondly, digital signatures can also be utilized in (entity) authentication protocols where the protocol is running in real-time and there is no meaning message other than the claim of being a particular entity [1].

With respect to both digital signatures and signature based authentication protocols, the use of public key cryptosystems raises the following problem [2]:

To verify the signature the public key of the signer is required. By the use of the public key it can be checked whether the signature was generated by the corresponding private key or not. However, it is not provable whether or not a certain entity owns the public key.

Obviously, the authentic link between the public key and its owner is needed. Most of the time this link is provided by public key certificates which are signed messages specifying an identity and the corresponding public key. Since each public key corresponds to a particular private key, a binding of the private key

to its owner is given indirectly. If the public key is not bound to its legal owner, attacks like the following become possible:

An adversary X generates his own key pair (PK_x, SK_x) and publishes the public key PK_x by claiming a wrong identity A. Afterwards, X is able to forge signatures of an entity A. This is because the verifier assumes that the public key used for the verification, belongs to its legal owner, namely entity A.

Public Key Infrastructure (PKI) provides protocols, services, and standards in order to employ public key certificates securely and effectively.

1.1 A New Player: Attribute Certificate

Public key certificates provide the best solution for most applications however some new applications like the ones in the area of health care require more than that [2]. Due to very dynamic rights of permissions and admissions, a certificate different than the public-key certificate is employed for professionals in healthcare and other similar organizational settings and is called attribute certificate.

Just like public-key certificates where an authentic link is established between an entity and his public key, attribute certificates link attributes to an entity in an authentic manner. An attribute certificate does not exist autonomously - it is rather bound to a base certificate: the public key certificate of the entity. Exemplarily, attributes describe some of the following characteristics: general authorizations, international or national specific data, delegations for other persons, temporary rights etc.

Similar to PKI, the attribute certificates framework defined by ITU provides a foundation upon which a Privilege Management Infrastructure (PMI) can be built.

When attribute certificates are also on the scene, the validation ¹ of the digital signature is slightly modified:

1. As seen from figure 1, the receiver needs to get four things to start the validation process: the signature, the message, the public key certificate and attribute certificate(s). Assume that the last two are obtained securely.
2. The receiver verifies the signature on the message by using the public key.
3. Based on the information gathered from the attribute certificate, the receiver decides if the signature owner is qualified to sign (when the signature is used for message authentication) or authorized to have an access (if the signature is used to authenticate the signature owner himself).

1.2 A Closer Look at PKI

In a PKI, the typical solution of the problem with public keys we have just introduced is to have trusted nodes known as certification authorities (CA) which

¹ We use the term validation to express something more than just simple mechanic verification of the signature.

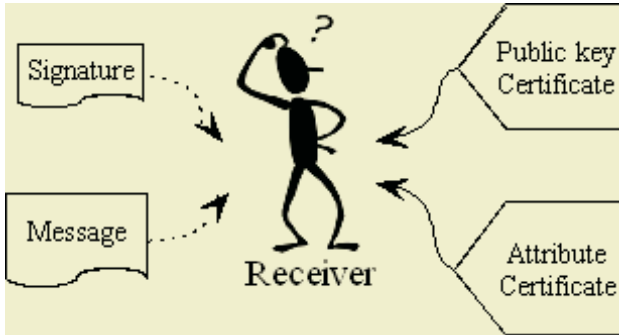


Fig. 1. Validation of Signature with Attribute Certificates

are *off-line* entities issuing public key certificates and putting them in any convenient location, such as the directory service. Being an offline entity, unlike the secret key equivalent of KDC (key distribution center) CA is neither a performance bottleneck nor a single point of failure; two parties (the signer and the verifier) can have a secure communication without a third party's involvement.

However there is potential problem with CAs. Consider an organization where all employees have certain authorizations. Suppose that an employee named Alice does something that warrants immediate revocation of her signing capability. For example Alice might be fired and since Alice is now a disgruntled ex-employee, it is required to alert others not to accept her signature as valid. One solution to this problem is to use a certificate revocation list (CRL) which lists the serial numbers of certificates that should not be honored. We have observed that CRLs and other similar approaches can be easily outdated (it is hard to update the list instantly) and are not the perfect answer to most organizations which need to provide more timely information about revocation status of employees. Current state-of-the-art approach is to use an on-line revocation server that can be queried over the network about the revocation status. For more details, please see the RFC of OCSP (Online Certificate Status Protocol) [4].

So in summary a typical PKI consists of (1) an offline CA (2) a directory service (3) an online revocation server. When the notion of PMI is first introduced, one obvious conventional solution is to use a similar three-component architecture for it. However attribute certificates have some unique characteristics to take into account before proposing a PMI design.

1.3 The Organization of the Paper

The rest of this paper is organized as follows. In the next section we explain the reasons to have two different (but linked) infrastructures and the differences between these two, PKI and PMI. In section 3, we summarize the previous work on PMI design and discuss their problems. We propose our new design in section 4. We show two different signature protocols suited to be used with our design in section 5. Section 6 is the conclusion.

2 PKI and PMI: Two-Egg Twins

An easy solution to incorporate attributes to the validation process is to include these attributes as extensions to a public key certificate and thus cancel separate attribute certificates by all means. However this solution would have several drawbacks:

1. **Different issuing authorities:** It is common in practice to have several independent authorities which issue either public key certificates or different kinds of attribute certificate. Developing such a distributed certificate management framework by employing only a single certificate is impractical if not impossible.
2. **Different validity periods:** In modern business, attributes of a person might change very frequently. If these attributes are included in a public key certificate, then it is required to set the validity period of this certificate to the shortest validity period of these attributes. Renewing certificates so frequently is really a burden for operation and management.
3. **Different revocation requirements:** As we have exemplified in subsection 1.2, it might be required to revoke certificates no matter how careful or pessimistic we are in setting the validity period. Looking at that example again, we see that actually the thing that should be revoked is the attribute of the employee which states that he is authorized to sign on behalf of that organization. If the public key certificate and attribute certificates are separated, there is no reason to revoke either the public key certificate or the other attribute certificates. Moreover, we notice that a public key certificate that binds only the identity of the person should be revoked very rarely (e.g. when the private key is stolen etc.)

Since attribute certificates and public key certificates have these aforementioned differences, it is much simpler to manage them by employing two different (but interlinked) infrastructures, PKI and PMI.

In this section, our initial aim was to answer two questions. Basically, to answer the first question regarding the necessity to have two different infrastructures, we have almost answered the second question about the differences between them. Let us make a few final comments:

The CAs in a PKI are generally external to the organization i.e. private companies or government agencies. This is because "identity" tends to have a global meaning. However attributes have a more local meaning e.g. being a project member. The PMI therefore can be managed locally and the issuing authority might be located inside the organization.

3 Previous Work on the Design of Privilege Management Infrastructure

As we already stated, a PMI design might be based on three distinct components as in PKI. Alternatively, in a recent paper [3], the authors proposed to

combine these three components and name it as Attribute Certificate Service Unit (ACSU) ². The ACSU is located inside the organization and composed of three components: (1) Attribute Authority (2) A database (directory) to store the attribute certificates (3) Attribute Server. To validate the signature, when a receiver queries ACSU for signer's attributes, ACSU returns back an attribute certificate only if it is not revoked.

This new design has several nice features:

1. Simplified design (revocation is a built-in element in the system).
2. Local control over employees' attribute certificates and their privileges.
3. Confidentiality of private privilege information.

Notice that the first feature exists because the components are combined and the last two are offered because the ACSU is located inside the organization.

3.1 The Problems in Dawson et al.'s Approach

In Dawson et al.'s approach when Bob receives a signature from Alice; to validate the signature, he gets her public key from a directory, he queries the revocation server to obtain the status of her public key certificate and then for authorization information he queries the ACSU to obtain her (unrevoked) attribute certificate(s). We have investigated the following problems in this design:

1. Although it is simpler than the conventional approach, Bob still needs to query two different servers to validate the signature.
2. Immediate revocation (fine grained control over signing capabilities) is not supported. The revocation server and ACSU can return the revocation information at the time of query not at the time of signing.

The first problem can be easily handled by covering the functionality of revocation server also inside the ACSU so that Bob queries only one server but the second problem needs a more careful treatment.

In a recent paper [5], the authors observe that traditional revocation techniques, including OCSP where the receiver of a signature asks a server for revocation information, do not provide immediate revocation.

In their paper they also propose a method to provide immediate revocation. However in their paper they do not discuss how attribute certificates can be managed. We will now discuss a new PMI design where attribute certificates are involved. But first we would like to extend a definition from which the concept of immediate revocation is derived:

Modified Definition of Binding Signature Semantics: A valid digital signature on a particular message can be generated only if its owner has a valid public key certificate and valid attribute certificate(s) (is authorized or qualified to sign that message) "at the time of signing".

² In their paper, they also propose a method to employ multiple ACSUs for scalability reasons.

3.2 When Do We Need Binding Signature Semantics?

Do we really need binding signature semantics? or it is sufficient to ensure that the certificates are not revoked at the validation time. The correct answer really depends on the application. In an application where signatures are used to authorize the authenticated entity to have an access to a resource in the system, the protocol works in real time in other words signing and validation are almost performed at the same time. In this situation "binding signature semantics" can be considered as a luxury. However it is not difficult to find an application that really needs such a fine grained control.

For instance, suppose that the second manager of a company is permitted to sign purchase orders when the first manager is on leave. Since the validation of the signature on the purchase orders can be done at some later time, it is vital to check the validity of certificates at the time of signing.

4 Our Proposal: Extending Fine-Grained Control with Attribute Certificates

Fine grained control can be easily provided by employing a semi-trusted online entity in signature generation. The ACSU-like server located inside the organization can act as this online server (Unlike our proposal in [5], this server and the CA were distinct entities). Now the ACSU is communicating with the signer not the verifier. There are various alternatives for implementation as we will see in the next section but the main idea is the same:

The server would participate in the signature generation only if the user is allowed to sign the message at hand (has valid public key and attribute certificates at the present time). Without server's participation there is no way for the user to generate a signature.

More precisely, as seen in Figure 2 the proposed PMI design involving attribute certificates works as follows:

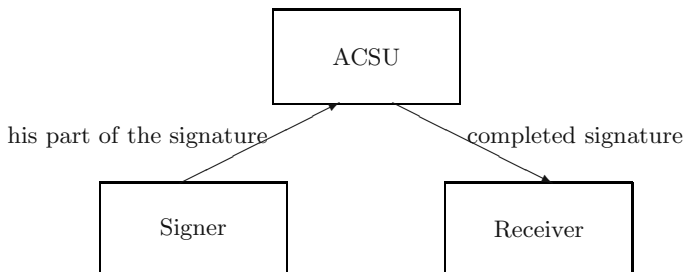


Fig. 2. Operation of the Proposed PMI Design

- The user produces his part on the signature of the message he would like to sign and sends it to the ACSU.
- ACSU checks whether the user's public key certificate is valid and also checks whether he has valid attribute certificate(s) to sign the message. If so, it generates its part of the signature on the message. Now the signature on the message is complete. It is sent to the intended receiver(s).
- The receiver verifies the signature on the message (Now the validation becomes much simpler and mechanic verification of the signature would be sufficient). He does not need to interact with anybody for verification.

5 Implementing Our New PMI Design

5.1 Threshold Signatures

In a (t, n) threshold signature scheme, the private key is shared between n users and at least t users out of n should collaborate to produce a valid digital signature. Any $(2, 2)$ threshold signature scheme where the signer himself and the ACSU constitute the two users would be sufficient to implement our new PMI design.

In [5], the authors proposed to use $(2, 2)$ threshold RSA (they call it mediated RSA) to realize their idea. We think they choose RSA [6] since unlike DSS [7] in RSA scheme it is very easy to share the key among two users and generate the signature accordingly.

5.2 Server Assisted One-Time Signatures

Mediated RSA serves the purpose most of the time, but for applications where users are holding constrained devices such as a handheld or a cell phone, performance of signature generation becomes an important criteria. Since most digital signatures depend on public key operations which are known to be very heavy, there are numerous work in the literature to employ a third party (a server) to offload some of the computation carried on users' constrained device.

In a conventional application where the CA is an offline entity and revocation is supported by CRLs, utilizing a third party beside the signer and receiver might be considered as an inconvenience. However we are now in a situation where a third party is a built-in element in the design. Then why not to use this third element (ACSU) for a second purpose, the purpose to improve the efficiency of signature generation?

As we have already stated, to realize this idea of "one stone-two birds" there are various alternatives. One promising example is SAOTS (Server Assisted One-Time Signature) [8] which enable the user to generate a standard public key signature without performing any public key operations at all. The user himself generates only a one-time signature that can be realized very efficiently since it is based on nothing more than a one-way function. For more details of this approach please consult the original paper [8].

6 Conclusion

In a recent paper [3], a new design of privilege management infrastructure (PMI) was proposed. In this proposal, to get the signer's attribute certificates the receiver of the signature queries an online server called ACSU, which is also the authority on issuing attribute certificates. In this paper, we present an alternative design where the signer not the receiver communicates with ACSU. In this design, the verifier can check the validity of the certificates (1) at the time of signing not at the time of receipt and (2) in a more convenient way without interacting with a server.

Our proposal here can be thought as the clever combination of clever ideas in two papers (Boneh et al's [5] and Dawson et al.'s [3]).

We also observe that when a server is a built-in element in the design, we can easily utilize this server for variety of purposes. One such usage is to improve the performance of signature generation for ordinary users. As a future work, we would like to investigate other opportunities for our new PMI design. One candidate is the integration with identity based techniques which was previously shown to be straightforward using a third party [9].

References

1. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone: Handbook of Applied Cryptography, CRC Press, 2001.
2. P. Wohlmacher and P. Pharow: Applications in Health Care using Public-Key Certificates and Attribute Certificates. 16th Annual Computer Security Applications Conference December 11-15, 2000, New Orleans, Louisiana.
3. E. Dawson, J. Lopez, J. A. Montenegro, E. Okamoto: A New Design of Privilege Management Infrastructure for Organizations Using Outsourced PKI. Information Security, 5th International Conference, ISC 2002 Sao Paulo, Brazil, September 30 - October 2, 2002. Lecture Notes in Computer Science 2433, Springer 2002.
4. M. Myers, R. Ankney, A. Malpani, S.Galperin, C. Adams: Internet public key infrastructure online certificate status protocol - OCSP. RFC 2560, June 1999.
5. D. Boneh, X.Ding, G. Tsudik: Fine-grained control of security capabilities, ACM Transactions on Internet Technology, Volume 4, Number 1, February 2004.
6. R.L. Rivest, A. Shamir, L.M. Adleman: A method for obtaining digital signatures and public key cryptosystems, Communications of the ACM, 21(2), 1978.
7. National Institute of Standards and Technology (NIST): FIPS Publication 186: Digital Signature Standard, May 19, 1994.
8. K. Bicaçci, N. Baykal: Server assisted signatures revisited, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004. Lecture Notes in Computer Science 2964, Springer 2004.
9. X. Ding, G. Tsudik: Simple Identity-Based Cryptography with Mediated RSA, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003. Lecture Notes in Computer Science 2612, Springer 2003.

How to Qualify Electronic Signatures and Time Stamps

Detlef Hühnlein

secunet Security Networks AG
Sudetenstraße 16
D-96247 Michelau, Germany
detlef.huehnlein@secunet.com

Abstract. In this work we will show how non-qualified electronic signatures and time stamps can be efficiently enhanced in order to equip them with similar features as qualified ones. In particular we will show how non-qualified electronic signatures can be used in business processes which require the written form. Furthermore we will show how to construct "interval-qualified" (IQ) time stamps which may serve as cost efficient alternative to qualified time stamps issued by a trusted authority. An IQ time stamp issued at time t_i is linked to two qualified time stamps issued at time T_1 and T_2 , in a way that one is able to prove that $T_1 < t_i < T_2$.

1 Introduction

Because of national implementations of [EC/1999/93, Article 5 Section 1 (a)], like [BGB, §126a] in Germany for example¹, electronic signatures which are based on a qualified certificate and created using a secure signature creation device, called *qualified electronic signatures* throughout this work, are deemed equivalent to handwritten signatures. This implies that business processes which traditionally require the written form, either need to apply paper or qualified electronic signatures in order to fulfill this formal requirement. As there are legal regulations, like [BGB, §125], which void statements lacking the appropriate form, this formal requirement turns out to be essential. While there are quite a few certification-service-providers in Europe issuing qualified certificates and secure signature creation devices, it is fair to say that Europe is still far away from the omnipresent availability of equipment to produce qualified electronic signatures. Hence there are situations, where the formal need for qualified electronic signatures introduces additional obstacles which may even lead to sticking with paper-based processes. In communities where non-qualified electronic signatures are already used, e.g. for electronic banking or e-mail protection, the formal need for *qualified* electronic signatures is even more annoying and it would be

¹ We will use the German law to exemplify the legal issues related to our proposal. These legal considerations may not translate directly, but would need to be compared to other national laws.

desirable to be able to "enhance the non-qualified electronic signatures in use", in order to make them equivalent to handwritten signatures.

In a similar fashion there is, due to operational necessities and legal requirements like [SigV, §17], an increasing demand for time stamps issued by a trusted authority. Following [SigG, §2 Nr. 14] we will call such time stamps which are produced in a trustworthy manner *qualified time stamps*. Because of costly requirements, such as the need for sophisticated security concepts, certified products and additional liability issues, it is not surprising that time stamping authorities typically charge relatively high fees per issued time stamp in order to realize a return on their initial investment. If a large number of time stamps is required, such fees may soon become a major cost factor and even jeopardize the business case of an electronic business process. On the other side, one can not simply ignore the costly trust issues and naively use self made time stamps, because they would provide far less evidence. Thus it would also be desirable to be able to "enhance self made time stamps" in a cost-efficient manner, such that they provide a similar level of evidence as a qualified time stamp.

In this work, we will propose simple solutions for these two problems: We show how to "qualify electronic signatures and time stamps" in a very cost efficient manner.

First we show how the combination of simple technical and legal measures can be used to enhance non-qualified electronic signatures such that they can be used for business processes which require the written form. As explained in Section 2, this is mainly achieved by having a (in a legal sense) properly authorized central signature server re-signing the non-qualified signatures of the clients using qualified signatures in an automated manner.

Next we show how a few qualified time stamps can be used to enhance an arbitrary number of self made time stamps with moderate computational cost. As explained in Section 3, this is mainly achieved by appropriately linking an arbitrary number of self made time stamps S_i , issued at time t_i , with two qualified time stamps issued at time T_1 and T_2 respectively, in a way that one is able to prove that $T_1 < t_i < T_2$ (see Theorem 1).

2 Qualifying Electronic Signatures

In this section we will show how a community which uses non-qualified electronic signatures may enhance these signatures, such that the signed documents fulfill the written form in the sense of [EC/1999/93, Article 5, Section 1 (a)].

Let N be a signatory, which is able to produce electronic signatures, denoted by $\sigma(x, N)$. However N is not able to produce qualified electronic signatures, because he does not use a secure signature creation device (SSCD) or he does not use a qualified certificate. Thus N could use software-based PGP-keys without certificates for example.

Suppose that N wants to declare its intention within a business process, for which the written form according to [BGB, §126] is required. Because of [BGB, §126 (3)], N will be able to use the electronic form according to [BGB,

§126a] instead of the written form. In order to fulfill the requirements stipulated there, N would need to use a qualified electronic signature according to [SigG, §2 Nr. 3] to sign a document D . Because N is not able to produce such a signature, it will need to make use of another signatory Q , which is able to produce qualified electronic signatures, denoted by $\sigma_q(x, Q)$. In practice, Q may be a server, which is able to create qualified electronic signatures in an automated manner as explained in [HüKn03].

In this scenario, as depicted in Fig. 1, N will send the electronic document D , which will need to be signed with a qualified electronic signature, together with an appropriate power of attorney P , according to [BGB, §167], to Q . P will contain an appropriate legal statement L_1 and the hash value $h(D)$ of the document D . L_1 states that Q is, for the special purpose necessary to proceed the document D , authorized to act on behalf of N . The power of attorney P will be signed by N with the non-qualified electronic signature $\sigma(P, N)$.

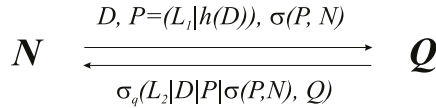


Fig. 1. How to qualify electronic signatures

Thus N will send the triple $D, P = (L_1|h(D)), \sigma(P, N)$ to Q . Q is verifying N 's signature and the content of the power of attorney P . If everything is ok, Q will produce the qualified electronic signature $\sigma_q(L_2|D|P|\sigma(P, N), Q)$. Here L_2 states that Q is presently not acting on behalf of its own, but on behalf of N . That Q is allowed to act on N 's behalf is shown by the power of attorney P . Note that according to [BGB, §179 (1)] Q can not be liable as long as it is acting within the scope of P . Finally the qualified electronic signature is returned to N .

Now the document D is, together with L_2 , P and $\sigma(P, N)$, signed by Q with a qualified electronic signature and hence fulfills the formal requirement of [BGB, §126a]. Because of L_2 , it is clear that Q is acting on behalf of N and that it is authorized to do so, because of P and $\sigma(P, N)$. Hence, as stipulated in [BGB, §164 (1)], the declaration of intention in D is as effective for or against N as it would have signed D itself.

The crucial point that this legal construction is possible, is that, according to [BGB, §167 (2)], the power of attorney P does not need to have the same form as the legal transaction for which it is intended. This implies that even if D requires a qualified electronic signature, it is sufficient that P is only signed with a non-qualified electronic signature.

3 Qualifying Time Stamps

In this section we will consider a similar scenario, in which N is able to issue time stamps, but N is not able to issue qualified time stamps according to

[SigG, §2 Nr. 14], because N is no certification-service-provider (CSP) which fulfills the costly requirements of [SigG, §§4-14 and §17 or §23]. As above, N would like to enhance its self made time stamps, such that they would provide a similar level of evidence as qualified time stamps issued by CSPs. This is realized by constructing "interval-qualified" (IQ) time stamps, which are linked to two qualified time stamps, such that one can prove that they must have been issued between them (see Theorem 1).

The function $h : \{0, 1\}^* \rightarrow \{0, 1\}^n$ is called a *cryptographic hash function*, if it is infeasible to find some preimage x given $h(x)$ (one-wayness) and that it is infeasible to find a pair x_1, x_2 such that $h(x_1) = h(x_2)$ (collision-resistance).

A *time stamp*, like specified in [RFC3161] for example, is roughly of the form

$$S = \sigma(h(D)|t, TSA), \quad (1)$$

where $h()$ is a cryptographic hash function, D is the data which is to be time stamped, t is the time and σ is a secure electronic signature scheme in the sense that only TSA is able to produce $\sigma(\cdot, TSA)$. A TSA is called *trusted* if it always inserts the present time in the time stamps it issues. A time stamp $QS = \sigma(\cdot, Q)$ is called a *qualified time stamp*² if Q is trusted.

The essential value of such a time stamp may be stated as follows.

Proposition 1. *If $QS = \sigma(h(D)|t, Q)$ is a qualified time stamp, then D existed prior to t .*

Proof: Because Q is trusted, QS proves that $h(D)$ has been existing at time t . Because of the one-wayness of $h()$, it is infeasible to find D given $h(D)$. Hence D must have been used to calculate $h(D)$, which implies that D existed prior to t . \square

Note that D is uniquely specified by $QS = \sigma(h(D)|t, Q)$ because of the collision-resistance of $h()$.

Next we will consider what happens, if one links time stamps as proposed in [HaSt90, Section 5.1] and [BLLV98, Lipm99].

Proposition 2. *Let $QS = \sigma(h(r)|T_1, Q)$ be a qualified time stamp and $S = \sigma(h(QS)|m|\tilde{t}, N)$ be a time stamp, where r and m are arbitrary data. Let t be the creation time of S . Then $t > T_1$.*

Proof: Because Q is trusted, the qualified time stamp QS is produced at time T_1 . Because of the one-wayness of $h()$ it is infeasible to find QS given $h(QS)$. Hence QS must have been used to calculate $h(QS)$, which implies that S was created at a later point in time than T_1 . \square

Note that t might be different from \tilde{t} , because N might not be trusted. The other way around is shown analogously.

² While the *technical* definition of a qualified time stamp given here, obviously differs from the *legal* definition given in [SigG, §2 Nr. 14], they tend to be "equivalent" in practice in the sense that until today all qualified time stamps, which meet the legal definition also meet the technical definition given here.

Proposition 3. Let $S = \sigma(h(r)|m|\tilde{t}, N)$ be a time stamp and $QS = \sigma(h(S)|T_2, Q)$ be a qualified time stamp, where r and m are arbitrary data. Let t be the creation time of S . Then $t < T_2$.

Proof. Because Q is trusted, the qualified time stamp QS is produced at time T_2 . Because of the one-wayness of $h()$ it is infeasible to find S given $h(S)$. Hence S must have been used to calculate $h(S)$, which implies that S was created prior to T_2 . \square

Combining these two simple results, we obtain the following corollary as visualized in Fig. 2.

Corollary 1. Let $QS_1 = \sigma(h(r)|T_1, Q)$ be a qualified time stamp, $S = \sigma(h(QS_1)|m|\tilde{t}, N)$ be a time stamp and $QS_2 = \sigma(h(S)|T_2, Q)$ be a qualified time stamp again, then $T_1 < t < T_2$, where t is the creation time of S .

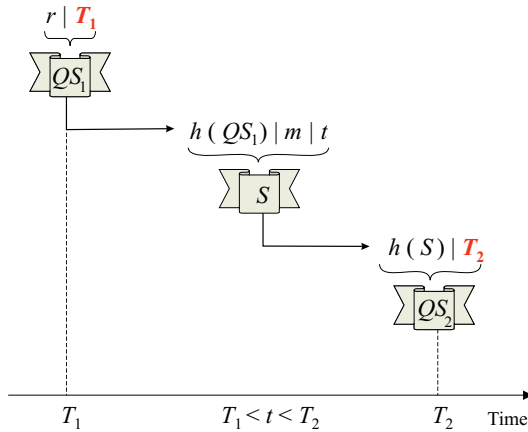


Fig. 2. Relative temporal order induced by $h()$

If there is more than one time stamp S_i in between the two qualified time stamps, then one will include QS_1 in every time stamp S_i , $0 < i < n$, and will link all these time stamps in an appropriate manner to the second time stamp QS_2 . For this purpose, one may use the batch signature strategy introduced in [PaBo99], which uses Merkle's authentication tree [Merk80] to construct an efficient batch signature scheme.

As shown in Fig. 3, one will construct a binary hash tree from the time stamps S_i and will obtain a qualified time stamp QS_2 for the root of this tree. The relation between S_i and QS_2 can be verified using the S_i -specific reduced hash tree, which consists of data necessary to reconstruct the path from the leaf S_i to the root node $R = h(\cdots h(S_i) \cdots)$, as explained in [PaBo99].

By combining the ideas from above, one obtains the IQ time stamps, consisting of the triple (QS_1, S_i, QS_2) , as shown in Fig. 4.

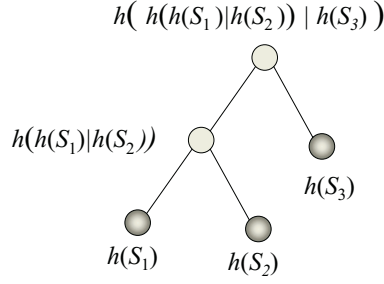


Fig. 3. Hash tree for batch signature with three nodes

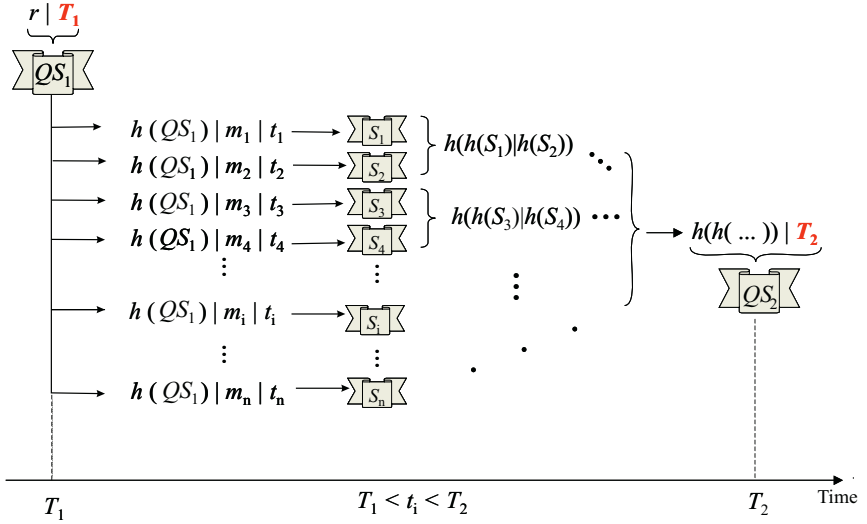


Fig. 4. Construction of interval-qualified time stamps

Theorem 1. Let $QS_1 = \sigma(h(r)|T_1, Q)$ be a qualified time stamp, $S_i = \sigma(h(QS_1) | m_i | \tilde{t}_i, N)$ be a time stamp and $QS_2 = \sigma(h(\dots h(S_i) \dots)|T_2, Q)$ be a qualified time stamp which is constructed as shown in Fig. 4, then $T_1 < t < T_2$, where t is the creation time of S_i .

Proof: $T_1 < t$ is shown in Proposition 2. That $t < T_2$ can be seen using the same argument for each application of $h()$ in the construction of the hash tree.

□

As depicted in Fig. 5, a typical IQ-timestamping system consists of *Clients*, an *Inhouse Timestamping Server (ITS)* and a *Cryptographic Service Provider (CSP)* and operates in the following steps:

1. The ITS requests a qualified time stamp QS_1 from the CSP.
2. The ITS issues an arbitrary number of time stamps S_i to the Clients, where each time stamp S_i includes the hash value of QS_1 .
3. Finally the ITS builds a hash tree from the time stamps S_i , as shown in Fig. 4, and obtains a qualified time stamp QS_2 for the root of this hash tree.

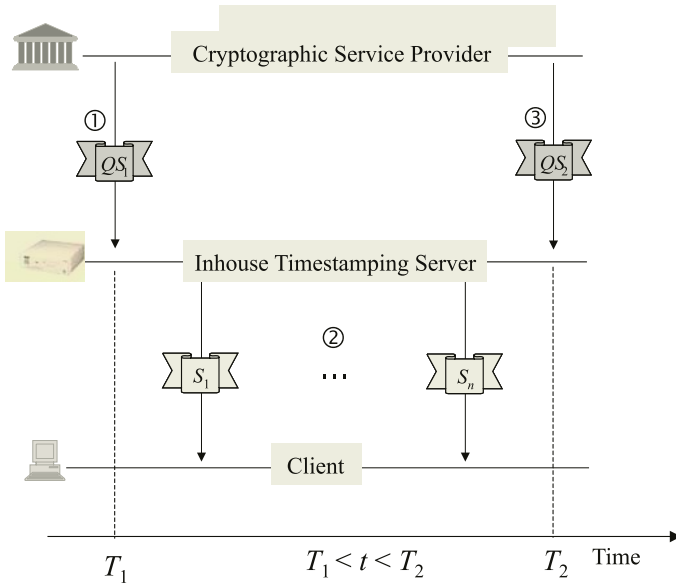


Fig. 5. IQ-timestamping system

4 Conclusion

In this work it was shown that it is possible to equip non-qualified signatures and time stamps with similar features as qualified ones. We believe that there are many application scenarios in which costly qualified electronic signatures and time stamps can be replaced by empowered signatures and IQ time stamps without a significant loss in quality. Hence, the simple ideas presented here may lead to more cost efficient solutions for electronic signatures and time stamps.

5 Acknowledgement

I would like to thank Ragna Tern and Volker Zeuner for fruitful discussions and the anonymous referees for valuable suggestions.

References

- BGB. *German Civil Code – Bürgerliches Gesetzbuch*, unofficial translation at <http://www.hull.ac.uk/php/lastcb/bgbengl.htm>
- BLLV98. Buldas, A.; Laud, P., Lipmaa, H., Villemson, J.: *Time-stamping with Binary Linking Schemes*, in *Advances in Cryptology – CRYPTO '98*, LNCS 1462, Springer-Verlag, 1998, pages 486-501
- EC/1999/93. *Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures*, via http://europa.eu.int/eur-lex/pri/en/oj/dat/2000/L_013/L_01320000119en00120020.pdf
- HüKn03. Hühnlein, D.; Knosowski, Y.: *Aspects of the automated generation of qualified electronic signatures*, in German, in *Proceedings of DACH 2003*, IT-Verlag, 2003, ISBN 3-00-010941-2, pages 293-307, via http://www.secunet.de/download/fachartikel/dach2003_aspekte-der-massensignatur.pdf
- HaSt90. Haber, S.; Stornetta, W.S.: *How to time-stamp a digital document*, in A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology – CRYPTO 90*, volume 537 of *Lecture Notes in Computer Science*, pages 437 - 455. Springer-Verlag, 1991
- Lipm99. Lipmaa, H.: *Secure and efficient time stamping systems*, PhD-thesis at the University of Tartu, Estonia, 1999, via <http://www.tcs.hut.fi/~helger/papers/thesis/thesis.pdf>
- Merk80. Merkle, R.: *Protocols for Public Key Cryptosystems*, *Proceedings of the 1980 IEEE Symposium on Security and Privacy* (Oakland, CA, USA, April 1980), pages 122–134
- PaBo99. Pavlovski C.; Boyd C.: *Efficient Batch Signature Generation using Tree Structures*, *International Workshop on Cryptographic Techniques and E-Commerce, CryptEC'99*, City University of Hong Kong Press, pages 70–77, via <http://sky.fit.qut.edu.au/boydc/papers/treefinal.ps>
- RFC3161. Adams, C.; Cain, P.; Pinkas, D.: *Internet X.509 Public Key Infrastructure - Time Stamp Protocol (TSP)*, RFC 3161, via <http://www.ietf.org>
- SigG. *Law Governing Framework Conditions for Electronic Signatures and Amending Other Regulations – Gesetz über Rahmenbedingungen für elektronische Signaturen und zur Änderung weiterer Vorschriften*, 16.05.2001, BGBl. 2001 Part I Nr. 22, page 876 ff, unofficial translation via <http://www.iid.de/iukdg/gesetz/Signaturg-engl.pdf>
- SigV. *Digital Signature Ordinance – Verordnung zur elektronischen Signatur*, 16.11.2001 BGBl. 2001 Part I Nr. 59, page 3074 ff, unofficial translation via <http://www.iid.de/iukdg/gesetz/SigV161101-engl.pdf>

An Efficient Revocation Scheme for Stateless Receivers^{*}

Yong Ho Hwang¹, Chong Hee Kim², and Pil Joong Lee^{1**}

¹ IS Lab., Dept. of Electronic and Electrical Eng., POSTECH, Korea
yhhwang@oberon.postech.ac.kr, pjl@postech.ac.kr

² Samsung Electronics Co., LTD, Korea
chonghee.kim@samsung.com

Abstract. A revocation scheme for stateless receivers enables the center to deliver information securely to the legitimate users over a public channel, where the receivers do not update their state from session to session. In this paper, we propose an efficient revocation scheme for stateless receivers. Our scheme uses a logical hierarchical key tree. Among previously proposed revocation schemes for stateless receivers, the SD scheme [10] is very efficient with respect to message length and Asano's schemes [1] are very efficient with respect to key storage. Using a binary key tree, our scheme has the same message length as the SD scheme and requires the storage of $\log n$ keys, whereas the SD scheme requires the storage of $\frac{1}{2} \log^2 n + \frac{1}{2} \log n + 1$ keys. Using an a -ary key tree, our scheme has the same key storage as Asano's method 2 and requires a message length of at most $2r - 1$, whereas Asano's schemes require a message length of at most $r(\frac{\log \frac{n}{r}}{\log a} + 1)$. Therefore, our scheme is more efficient than the SD scheme and Asano's schemes.

1 Introduction

Broadcast encryption schemes (or revocation schemes) enable a center to broadcast encrypted data to a large group of users, where only legitimate users can decrypt the data. Because the group of legitimate users is dynamically changing, the center should efficiently deliver available information to the group of legitimate users and prevent the group of revoked users from decrypting transmitted data.

An interesting variant of the broadcast encryption scheme is a revocation scheme for stateless receivers. A stateless receiver is a device that cannot update its original state, i.e. it is not capable of recording the past history of transmissions and changing its state accordingly. In a revocation scheme for stateless receivers, each receiver's operation is based on the current transmission and its

^{*} This research was supported by University IT Research Center Project and the Brain Korea 21 Project.

^{**} On leave at KT Research Center.

initial configuration. A revocation scheme for stateless receivers has many applications, such as pay-TV, the distribution of copyrighted materials, internet multicasting of video, music, and magazines, and so on.

A. Fiat and M. Naor first formalized the basic definitions and paradigms of the broadcast encryption scheme [6]. Afterwards, many variants have been investigated [5,4,7,11,12,13,14,3,9,16,17]. D.M. Wallner *et al.*[16] and C.K. Wong *et al.*[17] independently introduced the efficient broadcast encryption scheme based on a logical key tree. These schemes require the storage of $\log n + 1$ keys at each receiver and a message length of $2r \log n$, where n is the total number of users and r is the number of revoked users. They assume that their systems use a binary key tree. Many schemes based on the logical key tree were later proposed.

Revocation schemes for stateless receivers have been actively investigated in the past few years. In 2001, D. Naor *et al.* [10] proposed two schemes, the CS (Complete Subtree) scheme and the SD (Subset Difference) scheme. These schemes are based on the *Subset-Cover* framework, which partitions the set of non-revoked users into several subsets. The CS scheme requires a message length of $r \log \frac{n}{r}$ and the storage of $\log n$ keys at the receiver. The SD scheme requires a message length of $2r - 1$ and the storage of $\frac{1}{2} \log^2 n + \frac{1}{2} \log n + 1$ keys at each receiver. The SD scheme is very efficient with respect to message length.

In 2002, D. Halevy and A. Shamir proposed the LSD (Layered Subset Difference) scheme, which improved the SD scheme with respect to the storage of keys at each receiver [8]. The LSD scheme reduces the number of keys by using the notion of “*layer*”. However, the message length of the LSD scheme was larger than that of the SD scheme. The LSD scheme requires $O(r)$ message length and $O(\log^{1+\epsilon} n)$ keys where $0 < \epsilon < 1$.

T. Asano proposed two schemes (we call them Asano 1 and Asano 2, respectively), which are efficient with respect to key storage at each receiver [1]. Asano 1 requires one key and Asano 2 requires $\log_a n$ keys at each receiver. In addition, all of the two schemes have a message length of $r(\frac{\log \frac{n}{a}}{\log a} + 1)$. They used an a -ary tree and the security of their scheme is based on an assumption related to the RSA cryptosystem[15].

Our results. We propose an efficient revocation scheme whose security is based on the security of the RSA cryptosystem [15]. Our aim is to design a scheme with minimal message length and reasonable key storage. The SD scheme is the most efficient scheme with respect to message length among previously proposed revocation schemes. While our scheme has the same message length as the SD scheme, it requires the smaller key storage. In addition, our scheme requires a similar key storage and processing complexity to Asano’s schemes which are very efficient with respect to key storage.

Our scheme uses a logical hierarchical key tree and can be applied to a -ary key tree. The following table compares our scheme with other schemes when they use a binary key tree.

	Message length	The number of keys at receiver
Asano 1 [1]	$r \log \frac{n}{r}$	1
Asano 2 [1]	$r \log \frac{n}{r}$	$\log n$
SD scheme [10]	$2r - 1$	$\frac{1}{2} \log^2 n + \frac{1}{2} \log n + 1$
Our scheme	$2r - 1$	$\log n$

Table 1. Comparison with other schemes (a binary key tree)

In Table 2, we compare our scheme with the Asano schemes when an a -ary key tree is used.

	Message length	The number of keys	Processing time
Asano 1 [1]	$r(\frac{\log \frac{n}{r}}{\log a} + 1)$	1	$O(2^{a-1} \log_a n)$
Asano 2 [1]	$r(\frac{\log \frac{n}{r}}{\log a} + 1)$	$\log_a n$	$O(2^{a-1})$
Our scheme	$2r - 1$	$\log_a n$	$O(2^{a-1} \log_a n)$

Table 2. Comparison with other schemes (an a -ary key tree)

As shown in Table 1 and Table 2, our scheme is more efficient than the SD scheme and Asano's schemes.

2 Revocation Scheme for Stateless Receivers

In a revocation scheme, a session key k is encrypted and broadcasted with the symmetric encryption of the “actual” message. Generally, the encryption of k is called an *enabling block*.

Let N be the set of all users, R the set of revoked users, and $N \setminus R$ the set of the remaining users when r receivers are revoked. We suppose that $|N| = n$ and $|R| = r$. The goal of the revocation scheme is to allow the users in $N \setminus R$ to decrypt the message broadcasted by the center, while users in R cannot decrypt it even if the revoked users conspire with one another.

The center divides $N \setminus R$ into disjoint subsets and assigns different subset-keys for each subset. Then it encrypts a session key k with the subset-keys and broadcasts them. After receiving the ciphertext, each user in $N \setminus R$ finds the subset to which he belongs and deduces the subset-key from his initial secret information. Finally, he decrypts the encrypted message.

A *Revocation scheme for stateless receivers* **RS** consists of three parts (**Setup**, **Enc**, **Dec**):

- **Setup**: The center sets up all the parameters of the scheme and generates the secret information I_u for each user u . Then it gives the secret information

I_u to each user. Note that for each subset S_i the user u belongs, he can deduce the subset-key for S_i from his secret information.

- **Enc:** Given a set R of revoked users, the center performs the following:

1. Randomly chooses a session key k .
2. Finds a partition of the users in $N \setminus R$ into disjoint subsets S_{i_1}, \dots, S_{i_m} . Let L_{i_1}, \dots, L_{i_m} be the subset-keys associated with the above subsets.
3. Encrypts K with L_{i_1}, \dots, L_{i_m} and broadcasts the following ciphertext:

$$< [i_1, \dots, i_m, E_{L_{i_1}}(K), \dots, E_{L_{i_m}}(K)], F_K(M) >$$

Where i_j is the information about the members of S_{i_j} , $E_{L_j}(\cdot)$ is a block encryption scheme to deliver the session key to users, and $F_K(\cdot)$ is a stream encryption of the “actual” message with the session key K . The portion in square brackets is called the *enabling block*.

- **Dec:** Given a ciphertext $< [i_1, \dots, i_m, C_1, \dots, C_m], C >$, the procedure for each user u is as follows:

1. Finds i_j such that $u \in S_{i_j}$.
2. Deduces the corresponding subset-key L_{i_j} from I_u .
3. Decrypts C_{i_j} using L_{i_j} to obtain the session k .
4. Decrypts C using K and output M .

If the user u belongs to R , then he cannot obtain the session K from his secret information.

To construct the revocation scheme for stateless receivers, we need a *subset algorithm* that divides $N \setminus R$ into disjoint subsets and a *key assignment method* that assigns subset-keys for each subset and the secret information for each user.

A key assignment method must satisfy the following properties:

Definition 1 *Correctness and Coalition-Resistance for a key assignment method*

- (1) **Correctness:** If a user u is not revoked, he should retrieve the subset-key for the subset to which he belongs only from the secret information I_u and the current transmission.
- (2) **Coalition-Resistance:** Even though all revoked users conspire with one another, it should be impossible to get a subset-key for any subset in the current transmission.

3 Proposed Scheme

We propose a new revocation scheme using a logical a -ary hierarchical key tree. It is secure under the assumption related to the RSA cryptosystem. In our scheme, each receiver has $\log_a n$ keys, and the message length is at most $2r - 1$.

The center imagines a rooted full a -ary tree T with n leaves (assume that n is a power of a) and assigns each user to each leaf of the tree. Each node is named v_i ($1 \leq i \leq n + \frac{n-1}{a-1}$). (We suppose that each node is named from leaves to the root, i.e. each user is named v_1, \dots, v_n and the root is named $v_{n+\frac{n-1}{a-1}}$).

3.1 Subset Algorithm

For a given set R of revoked users, the subset algorithm finds a collection of disjoint subsets S_{i_1}, \dots, S_{i_m} , which partitions $N \setminus R$. Let v_{j-l} be the l -th child node of v_j , and T_i a subtree rooted at node v_i . In our algorithm, if all the leaves of T_i are revoked, then v_i is regarded as a revoked node.

We define a half-revoked node as follows.

1. Let $v_{j,b_1b_2\dots b_a}$ denote the half-revoked node v_j , where $b_l = \{0, 1\}$, $\sum_{l=1}^a b_l \neq 0$ or a , and b_l is the status of v_{j-l} . If $b_l = 1$, then v_{j-l} is a revoked node. Otherwise, v_{j-l} is a non-revoked node.
2. If $b_l = 0$, then all leaves of T_{j-l} are non-revoked leaves.
3. A half-revoked node has at least one revoked child.

Fig. 1 is an example of the half-revoked node.

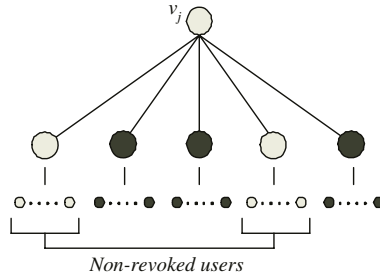


Fig. 1. A half-revoked node v_j (all black leaves are revoked users)

Let b be $b_1b_2\dots b_a$. In the case of Fig. 1, $v_{j,b}$ is denoted by $v_{j,b} = v_{j,01101}$. Then a subset is represented by two nodes $(v_i, v_{j,b})$ such that v_i is the ancestor of $v_{j,b}$ or $v_i = v_{j,b}$. We denote such a subset by $S_{i,(j,b)}$. A user is in $S_{i,(j,b)}$ if and only if he is a leaf of T_i but not a leaf of T_{j-l} where $b_l = 1$. Fig. 2 depicts $S_{i,(j,b)}$.

Subset algorithm of the proposed scheme:

1. Consider all nodes whose depth is 1. Among them, find revoked and half-revoked nodes.
Remove all revoked leaves from T .
2. Iterate the following steps from $x = 2$ to d . ($d = \log_a n$)
 - (a) Consider children of v_i whose depth is x
 - (b) Count the number of revoked children.
Let this number be s .
 - (c) Count the number of children who have a half-revoked descendent.
Let this number be m .

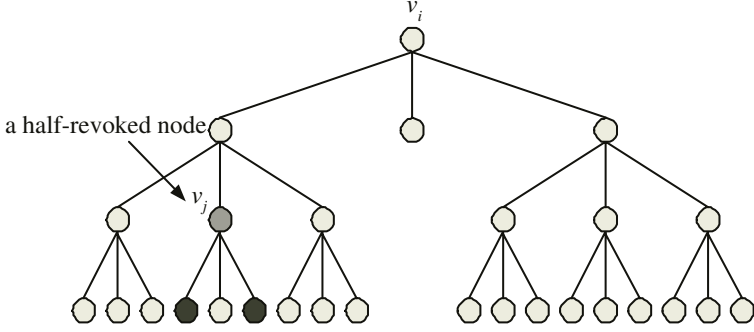


Fig. 2. the $S_{i,(j,b)}$ contains all non-black leaves. (A subset only has one half-revoked node)

- (d) If $m \neq 0$ and $m + s > 1$, then
 - i. Add m subsets, $S_{i_1,(j_1,b_1)}, \dots, S_{i_m,(j_m,b_m)}$.
(Where $v_{i_h} (1 \leq h \leq m)$, a child of v_i , is a half-revoked node or a node having a half-revoked descendent. If v_{i_h} is a half-revoked node, $i_h = j_h$. Otherwise, v_{j_h} , a descendent of v_{i_h} is a half-revoked node.)
 - ii. If $m + s < a$, then revoke $v_{i_h} (1 \leq h \leq m)$, make v_i be a half-revoked node $v_{i,b}$, and remove T_{i-l} from T where $b_l = 1$.
 - iii. If $m + s = a$, then revoke v_i and remove all descendants of v_i from T .
- (e) If $m = 0$ and $s > 0$, then make v_i be a half-revoked node $v_{i,b}$ and remove T_{i-l} from T where $b_l = 1$
- (f) If $d = x$, and $v_{n+\frac{n-1}{a-1}}$ is a half-revoked node or a node having a half-revoked descendent, then add one subset $S_{n+\frac{n-1}{a-1},(j_h,b_h)}$ where v_{j_h} is a half-revoked node.

Fig. 3 provides an example to explain our subset algorithm. In Fig. 3, if the black marked users are revoked, then the half-revoked nodes $v_{28,100}$, $v_{30,101}$ are made in Step 1 of the subset algorithm. In Step 2, two subsets $S_{28,(28,100)}$, $S_{30,(30,101)}$ are added and v_{37} is regarded as a half-revoked node $v_{37,101}$. In addition, a subset $S_{40,(37,101)}$ is added in the last iteration of Step 2. Therefore, the process generates a total of 3 subsets, $S_{28,(28,100)}$, $S_{30,(30,101)}$, and $S_{40,(37,101)}$.

Theorem 1 *Given any set of revoked receivers R , the proposed subset algorithm partitions $N \setminus R$ into at most $2r-1$ disjoint subsets when it uses an a -ary key tree.*

Proof. In Step 1 of our subset algorithm, because a half-revoked node has at least one revoked leaf, the number of revoked and half-revoked nodes is smaller than that of revoked leaves. In Step 2.(d).i, if a node v_i has m half-revoked nodes and s revoked children, then our subset algorithm adds $m (\leq k)$ subsets where $k = m + s$. If $k = a$, then v_i is revoked and if $1 < k < a$, then $v_{i,b}$ is regarded as

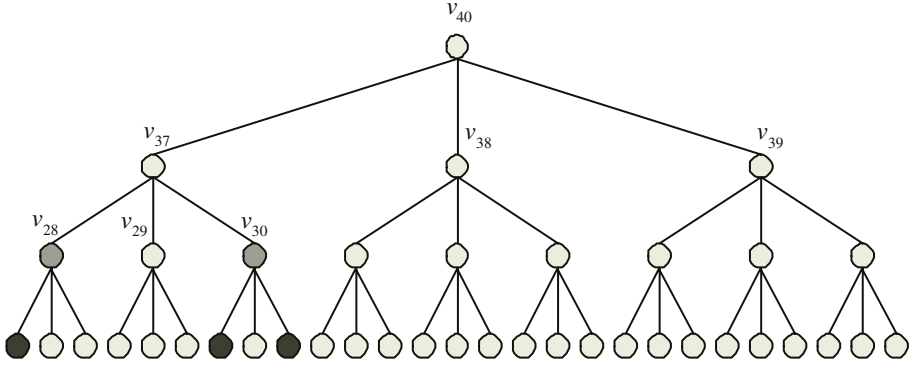


Fig. 3. Example for our subset algorithm

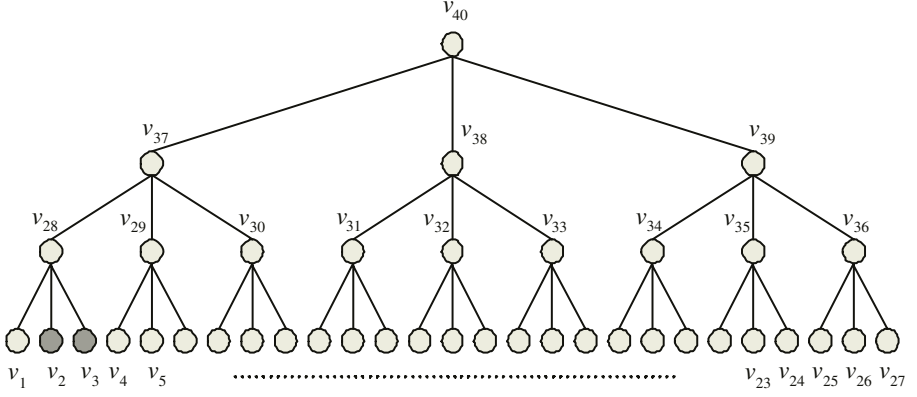
a half-revoke node. Afterwards, $T_{i-l}(1 \leq h \leq m)$ is removed from T where $b_l = 1$. In this case, the number of revoked and half revoked nodes is reduced by $k - 1$ and m subsets are added. In the case of Step 2.(e), a subset is not added, and the number of revoked and half-revoked nodes in T is reduced. Therefore, when the number of revoked and half-revoked nodes in T is reduced by at least $k - 1$, at most k subsets are added. Here, the worst case is that $k = 2$. When $k = 2$, the number of revoked and half-revoked nodes reduces by 1 and 2 subsets are added. However, if $k = 1$ in the last iteration of Step 2, then only one subset is added in Step 2.(f). Consequently, the process generates a total of $2r - 1$ ($= 2(r - 1) + 1$) subsets. (If the system uses a binary key tree, then the subsets of our scheme are the same as those of the SD scheme.)

Moreover, every non-revoked user v_q is in exactly one subset. Assume that v_q is in u disjoint subsets $S_{i_1, (j_1, b_1)}, \dots, S_{i_m, (j_m, b_d)}$. We choose two subset, $S_{i_l, (j_l, b_l)}, S_{i_k, (j_k, b_k)}$ among m subsets. If the depth of v_{i_l} is larger than that of v_{i_k} , then v_{i_l} is an ancestor of v_{i_k} because v_{i_l} and v_{i_k} are ancestors of v_q . When $S_{i_k, (j_k, b_k)}$ is added in Step 2.(d).i, v_{i_k} is revoked in Step 2.(d).ii. Therefore, when $S_{i_l, (j_l, b_l)}$ is added, v_{i_k} is regarded as a revoked node and the descendants of v_{i_k} are not included to $S_{i_l, (j_l, b_l)}$. Then, v_q is not included to $S_{i_l, (j_l, b_l)}$ because v_q is a descendant of v_{i_k} . However, this is contradictory, hence the assumption that v_q is in u disjoint subsets is wrong. \square

3.2 Key Assignment Method

Setup. The center chooses two large primes p, q and publishes M ($= pq$). The two primes p, q are kept secret. The center randomly chooses and assigns K_i ($\in Z_M^*$) to the root v_i of the subtree T_i ($n + 1 \leq i \leq n + \frac{n-1}{a-1}$). The total number of K_i is $\frac{n-1}{a-1}$. The center generates and publishes $(2^a - 2)\frac{n-1}{a-1}$ primes $p_{j, b_1 b_2 \dots b_a}$ (such that $\gcd(\phi(M), p_{j, b_1 b_2 \dots b_a}) = 1$) and W_j , where $j = n + 1, \dots, n + \frac{n-1}{a-1}$ and

W_i is a product of all the primes assigned to the subtree T_i . Fig. 4 depicts an example of our key assignment method for $n=27$ and $a=3$.



The published primes: $W_i, P_{i,101}, P_{i,010}, P_{i,011}, P_{i,100}, P_{i,101}, P_{i,110}$ ($i=28, \dots, 40$)

The assigned keys each node: K_i ($i=28, \dots, 40$)

Fig. 4. An example of our scheme for $n=27$ and $a=3$

The subset-key $L_{i,(j,b)}$ for the subset $S_{i,(j,b)}$ is $H(I_{i,(j,b)})$, where $H(\cdot): \{0,1\}^* \rightarrow \{0,1\}^l$ is a one-way hash function and $I_{i,(j,b)}$ is computed as follows:

$$I_{i,(j,b)} = K_i^{\frac{W_i}{P_{j,b}}} \bmod M$$

If v_{j-i} is revoked, then $b_i = 1$ and otherwise $b_i = 0$. For example, suppose that $User_2$, $User_3$, and $User_{23}$ are revoked in Fig. 4. Then the remaining set is divided into tree subsets, $S_{37,(28,011)}$, $S_{39,(35,010)}$ and $S_{40,(40,101)}$. The subset-keys, $L_{37,(28,011)} = H(I_{37,(28,011)})$, $L_{39,(35,010)} = H(I_{39,(35,010)})$ and $L_{40,(40,101)} = H(I_{40,(40,101)})$, where $I_{37,(28,011)} = K_{37}^{\frac{W_{37}}{P_{28,011}}} \bmod M$, $I_{39,(35,010)} = K_{39}^{\frac{W_{39}}{P_{35,010}}} \bmod M$ and $I_{40,(40,101)} = K_{40}^{\frac{W_{40}}{P_{40,101}}} \bmod M$. For convenience, we will omit “mod M ” in the remainder of this paper where it is clear from the context.

Next we describe the information I_u that each receiver v_u receives in order to derive the subset-key. Let v_{k-l} be a node between v_i and v_u , and $Path_{i,u}$ the product of $p_{k,b}$ primes where $b_l = 1$. Then, for each subtree T_i such that v_u is a leaf of T_i , $K_i^{Path_{i,u}}$ is the secret information for v_u . In Fig. 4, the secret information for $User_1$ are $(K_{40}^{Path_{40,1}}, K_{37}^{Path_{37,1}}, K_{28}^{Path_{28,1}})$ where $Path_{40,1} = p_{40,100} p_{40,101} p_{40,110} p_{37,100} p_{37,101} p_{37,110} p_{28,100} p_{28,101} p_{28,110}$, $Path_{37,1} = p_{37,100} p_{37,101} p_{37,110} p_{28,100} p_{28,101} p_{28,110}$ and $Path_{28,1} = p_{28,100} p_{28,101} p_{28,110}$. Each subtree T_i that contains v_u contributes one key from the center, therefore, the total number of keys stored by receiver v_u is $\log_a n$.

Enc. The center divides the subsets and computes the subset keys. Afterwards, he encrypts a session key k with the subset keys and broadcasts a ciphertext. Suppose that $User_2$, $User_3$, and $User_{23}$ are revoked in Fig. 4. The center encrypts the session k with $L_{37,(28,011)} = H(I_{37,(28,011)})$, $L_{39,(35,010)} = H(I_{39,(35,010)})$ and $L_{40,(40,101)} = H(I_{40,(40,101)})$ and broadcasts the following ciphertext:

$$\begin{aligned} &< [i_1, i_2, i_3, E_{L_{i_1}}(k), E_{L_{i_2}}(k), E_{L_{i_3}}(k)], F_k(M) > \\ \text{where } i_1 &= \{37, (28, 011)\}, i_2 = \{39, (35, 010)\}, i_3 = \{40, (40, 101)\} \end{aligned}$$

Dec. When a non-revoked user v_u receives the ciphertext, he finds the subset $S_{i,(j,b)}$ that includes him and computes the subsets keys as follows:

$$L_{i,(j,b)} = H(I_{i,(j,b)}) \text{ where } I_{i,(j,b)} = (K_i^{Path_{i,u}})^{\frac{W_i}{p_{j,b}^{Path_{i,u}}}} \quad (1)$$

If no users in N are revoked, the subset key K_i is $L_i = H(K_i^{W_i})$ where i is $n + \frac{n-1}{a-1}$. In the above example, when the non-revoked user, $User_1$, receives the above ciphertext, he can deduce $L_{37,(28,011)}$ from his secret information as follows:

Given his secret information $(S_1, S_2, S_3) = (K_{40}^{Path_{40,1}}, K_{37}^{Path_{37,1}}, K_{28}^{Path_{28,1}})$, $User_1$ computes $L_{37,(28,011)} = H(S_2^{\frac{W_{37}}{p_{28,011}^{Path_{37,1}}}})$, where

$$\begin{aligned} &\frac{W_{37}}{p_{28,011}^{Path_{37,1}}} \\ &= \frac{p_{37,001}p_{37,010}p_{37,011}p_{37,100}p_{37,101}p_{37,110}p_{28,001}p_{28,010}p_{28,011}p_{28,100}p_{28,101}p_{28,110}W_{29}W_{30}}{p_{28,011}p_{37,100}p_{37,101}p_{37,110}p_{28,100}p_{28,101}p_{28,110}} \\ &= p_{37,001}p_{37,010}p_{37,011}p_{28,001}p_{28,010}W_{29}W_{30}. \end{aligned}$$

However, revoked users, $User_2$ and $User_3$ cannot compute $L_{37,(28,011)}$ from their secret information $K_{37}^{Path_{37,2}}$ and $K_{37}^{Path_{37,3}}$ because they cannot compute the inverse of $p_{28,011}$.

3.3 Security of the Proposed Scheme

To discuss the security of our scheme, we first state the notion of the security of a revocation scheme. To say that an adversary has broken the system means that the adversary can obtain some information about the subset key or the encrypted message when all revoked users have provided their secret information. Therefore, we will show that the proposed scheme is secure against any collusion of revoked users under the following facts related to the RSA cryptosystem.

Assumption 1 [1] *If factors q_1, q_2 of a large composite $M = q_1q_2$ are unknown then computing p^{th} roots (mod M) for integral $p > 1$ is difficult.*

We introduce a theorem and a corollary proven in the appendix to [2].

Theorem 2 [1] *Let t and t_1, \dots, t_n be given integers and suppose there is a computable function F for which $K^t = F(K^{t_1}, K^{t_2}, \dots, K^{t_n}) \bmod M$ for every $K \in Z_M^*$, the group of units mod M . Let $d = \gcd\{t_i\}$, $e = \gcd\{t, d\}$, and $p = \frac{d}{e}$. Then, we can compute p^{th} roots in Z_M^* .*

Suppose that such a function F exists for $p > 1$, then we can compute non-trivial p^{th} roots in Z_M^* . This is contradictory to Assumption 1. Therefore, the following corollary holds.

Corollary 3 [1] *Under Assumption 1, such function exists only for $p=1$, namely $\gcd\{t_i\}|t$.*

At this point, we show that our scheme is secure by Assumption 1, and Corollary 3.

Theorem 4 *Our scheme is secure against any collusion of revoked users under Assumption 1 related to the RSA cryptosystem.*

Proof. Let \mathbf{G} be a set of secret keys of revoked users, which are the integer power of $K_i \bmod M$, $\{K_i^{t_1} \bmod M, \dots, K_i^{t_k} \bmod M\}$. Suppose (i) a function F exists to compute for a subset-key, $I_{i,m} = K_i^{\frac{W_i}{p_m}} = K_i^{t_m} \bmod M$ from the set \mathbf{G} and (ii) $I_{i,m}$ is not included in any key of \mathbf{G} .

On one hand, from (i) and the Corollary 3,

$$\gcd\{t_i\}|t_m \quad (2)$$

must hold. By definition, t_i is a product of primes assigned to revoked nodes. Therefore, we can write $t_i = \alpha_i p_m$ where $\gcd(\alpha_i, p_m) = 1$ from (ii) and $\gcd\{t_i\} = \alpha p_m$ where $\gcd(\alpha, p_m) = 1$. On the other hand, since W_i is a product of all primes assigned to T_i , we can write $W_i = \beta p_m$ where $\gcd(\beta, p_m) = 1$ (i.e. $\beta = t_m$) and then $\alpha p_m || \beta$ from (2). However, $\gcd(\alpha, p_m) = \gcd(\beta, p_m) = 1$. This is contradictory, for the assumption that I_m is not included in any key of \mathbf{G} and can be derived from \mathbf{G} is incorrect. \square

In Fig. 4, a user included in the subset $S_{40,(40,101)}$ can compute the subset key $L_{40,(40,101)}$ from his secret information by (1). However, while the users excluded from the subset $S_{40,(40,101)}$ can collude to obtain $K_{40}^{\frac{W_{40}}{p_{40,101}}}$, they cannot deduce information on $K_{40}^{\frac{W_{40}}{p_{40,101}}}$ because $\gcd(\frac{W_{40}}{p_{40,101}}, p_{40,101}) = 1$ and $p_{40,101} | \gcd(\text{Path}_{40,1}, \dots, \text{Path}_{40,9}, \text{Path}_{40,19}, \dots, \text{Path}_{40,27}) \rightarrow \gcd(\text{Path}_{40,1}, \text{Path}_{40,2}, \dots, \text{Path}_{40,9}, \text{Path}_{40,19}, \dots, \text{Path}_{40,27}) \nmid \frac{W_{40}}{p_{40,101}}$.

For a detailed example, suppose that $User_2$, $User_3$, and $User_{23}$ are revoked in Fig. 4. Then the revoked users' secret information is shown as follows:

Suppose that $User_2$ and $User_3$ conspire with each other. Let $K_i^{t_1} = K_{37}^{\text{Path}_{37,2}}$, $K_i^{t_2} = K_{37}^{\text{Path}_{37,3}}$ and $K_i^t = I_{37,(28,011)} = K_{37}^{\frac{W_{37}}{p_{28,011}}}$. Then $t_1 = \text{Path}_{37,2} = p_{37,100} p_{37,101} p_{37,110} p_{28,010} p_{28,011} p_{28,110}$, $t_2 = \text{Path}_{37,3} = p_{37,100} p_{37,101} p_{37,110} p_{28,001} p_{28,011} p_{28,101}$, and $t = \frac{W_{37}}{p_{28,011}} = p_{37,001} p_{37,010} p_{37,011} p_{37,100} p_{37,101} p_{37,110} p_{28,001} p_{28,010} p_{28,100} p_{28,101} p_{28,110} W_{29} W_{30}$. Therefore, $\gcd(t_1, t_2) = p_{37,100} p_{37,101} p_{37,110} p_{28,011}$, which does not divide $t = p_{37,001} p_{37,010} p_{37,011} p_{37,100} p_{37,101} p_{37,110} p_{28,001} p_{28,010} p_{28,100} p_{28,101} p_{28,110} W_{29} W_{30}$.

Revoked users	Secret information
$User_2$	$K_{40}^{Path_{40,2}}, K_{37}^{Path_{37,2}}, K_{28}^{Path_{28,2}}$
$User_3$	$K_{40}^{Path_{40,3}}, K_{37}^{Path_{37,3}}, K_{28}^{Path_{28,3}}$
$User_{23}$	$K_{40}^{Path_{40,23}}, K_{39}^{Path_{39,23}}, K_{35}^{Path_{35,23}}$

Table 3. Example of the secret information for each user in the proposed scheme

4 Discussion on the Proposed Scheme

We compare our scheme with other efficient schemes, the SD scheme and Asano's schemes. The SD scheme [10] is very efficient with respect to message length and Asano's schemes [1] are very efficient with respect to key storage among previously proposed schemes.

When our scheme uses a binary hierarchical key tree, namely $a=2$, the subsets of our scheme are the same as those of the SD scheme [10]. Therefore, the message length of our scheme is at most $2r - 1$. In actuality, the secret keys for each user of our scheme are elements in Z_M^* . Therefore, the size of each key is 8 times larger than that of the SD scheme [10]. (We suppose that $|M| = 1024$ bits in ours and the bit length of the secret key of the SD scheme is 128bits). However, our scheme is more efficient than the SD scheme with respect to the storage at each receiver. The message length of our scheme is also the same as that of the SD scheme. The following table shows the comparison of the SD scheme and our scheme.

	Actual storage of keys at receiver	
	SD scheme / Our scheme	Bits ($n = 2^{24} \approx 1,700,000$)
SD scheme[10]	$\frac{1}{16} \log n$	38,528 bits
Our scheme	1	24,576 bits

Table 4. Comparison with the SD scheme

Even if our scheme uses an a -ary hierarchical key tree, the extended subset algorithm of the SD scheme can be applied to our scheme. Let a half-revoked node in our scheme be regarded as a revoked node in the SD scheme. Then we can iteratively construct the subsets from the subset algorithm of the SD scheme. In the subset algorithm of the SD scheme, if the least-common-ancestor of two revoked nodes, v_l and v_k , is v_i , then two subsets $S_{i,l}$ and $S_{i,k}$ are made and v_i is revoked. Therefore, when the number of revoked nodes reduces by one, two subsets are made. Consequently, the number of subsets in the SD scheme is $2r - 1$. In our scheme, if v_i has k half-revoked descendants and revoked children, v_{l_1}, \dots, v_{l_k} ($1 \leq k \leq a$), then at most k subsets, $S_{i,(v_{l_1},b)}, \dots, S_{i,(v_{l_k},b)}$ are added, and v_i is revoked or half-revoked. When the number of revoked and half-revoked

nodes reduces by $k - 1$, k subsets are made. The worst case is when $k = 2$. When $k = 2$, the number of revoked and half-revoked nodes reduces by 1, and 2 subsets are made. Therefore, the number of subsets in our scheme is at most $2r - 1$ even if our scheme uses an a -ary key tree. We compare our scheme with Asano's schemes in Table 5. Here, we assume that $n = 2^{24}$, $r = 2^{12}$, $a = 5$.

	Actual number of subsets (Asano's scheme/Our scheme)	Enabling block length
Asano's scheme[1]	$\frac{\log \frac{N}{r}}{2 \log a} \approx 3$	3,670,016 bits
Our scheme	1	1,048,576 bits

Table 5. Comparison with the Asano schemes

5 Conclusion

We proposed an efficient revocation scheme for stateless receivers. When our scheme uses a binary key tree, it has the same message length as the SD scheme and requires the storage of $\log n$ keys, whereas the SD scheme requires the storage of $\frac{1}{2} \log^2 n + \frac{1}{2} \log n + 1$ keys. When our scheme uses an a -ary key tree, it has the same key storage as Asano 2 and requires a message length of at most $2r - 1$, whereas Asano's schemes require a message length of at most $r(\frac{\log \frac{N}{r}}{\log a} + 1)$. Therefore, our scheme has the minimal message length and the reasonable key storage, and is efficiently applied to practical systems in various environments.

References

1. T. Asano, A revocation scheme with minimal storage at receivers, *ASIACRYPT'02*, LNCS V.2501, pp.433-450, 2002.
2. S.G. Akl and P.D. Taylor, Cryptographic solution to a problem of access control in a hierarchy, *ACM Transactions on Computer Systems*, V.1. No. 3, pp.239-248, 1983.
3. D. Boneh and M. Franklin, An efficient public key traitor tracing scheme, *CRYPTO'99*, LNCS V.1666, pp.338-353, 1999.
4. D. Boneh and J. Shaw, Collusion-secure fingerprinting for digital data, *IEEE Transaction on Information Theory* 44(5), pp.1897-1905, 1998.
5. B. Chor, A. Fiat, and M. Naor, Tracing traitor, *CRYPTO'94*, LNCS V.839, pp.257-270, 1994.
6. A. Fiat and M. Naor, Broadcast encryption, *CRYPTO'93*, LNCS V.773 pp.480-491, 1993.
7. E. Gafni, J. Staddon, and Y.L. Yin, Efficient methods for integrating traceability and broadcast encryption, *CRYPTO'99*, LNCS V.1666, pp.372-287, 1999.
8. D. Halevy and A. Shamir, The LSD broadcast encryption scheme, *CRYPTO'02*, LNCS, V.2442, pp.47-60, 2002.

9. K. Kurosawa and Y. Desmedt, Optimum traitor tracing and asymmetric schemes, *EUROCRYPT'98*, LNCS V.1403, pp.145-157, 1998.
10. D. Naor, M. Naor, and J. Lostpiech, Revocation and tracing schemes for stateless receivers, *CRYPTO'01*, LNCS V.2139, pp.41-62, 2001.
11. M.Naor and B. Pinkas, Threshold traitor tracing, *CRYPTO'98*, LNCS V.1462, pp.502-517, 1998.
12. B. Pfitzmann, Trials of traced traitors, *Workshop on Information Hiding*, LNCS V.1174, pp.49-64, 1996.
13. B. Pfitzmann and M. Waidner, Asymmetric fingerprinting for large collusions, *ACM conference on Computer and Communication Security*, pp.151-160, 1997.
14. D.R. Stinson and R. Wei, Combinatorial properties and constructions of traceability schemes and frameproof codes, *SIAM Journal on Discrete Math* 11(1), pp.41-53, 1998.
15. R.L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, V.21. pp.120-126, 1978.
16. D.M. Wallner, E.J. Harder, and R.C. Agee, Key management for multicast: Issues and Architectures, *IETF Network Working Group*, RFC 2627, 1999.
17. C.K. Wong, M. Gouda, and S. Lam, Secure group communications using key graphs, *ACM SIGCOMM'98*, pp.68-79, 1998.

On the Use of Weber Polynomials in Elliptic Curve Cryptography^{*}

Elisavet Konstantinou^{1,2}, Yannis C. Stamatiou^{1,3,4}, and Christos Zaroliagis^{1,2}

¹ Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece

² Dept of Computer Eng. & Informatics, Univ. of Patras, 26500 Patras, Greece
`{konstane,zaro}@ceid.upatras.gr`

³ Dept of Mathematics, Univ. of the Aegean, Karlovassi, 83200, Samos, Greece
`stamatiu@aegean.gr`

⁴ Joint Research Group (JRG) on Communications and Information Systems
Security (Univ. of the Aegean and Athens Univ. of Economics and Business)

Abstract. In many cryptographic applications it is necessary to generate elliptic curves (ECs) with certain security properties. These curves are commonly constructed using the *Complex Multiplication* method which typically uses the roots of *Hilbert* or *Weber* polynomials. The former generate the EC directly, but have high computational demands, while the latter are faster to construct but they do not lead, directly, to the desired EC. In this paper we present in a simple and unifying manner a complete set of transformations of the roots of a Weber polynomial to the roots of its corresponding Hilbert polynomial for all discriminant values on which they are defined. Moreover, we prove a theoretical estimate of the precision required for the computation of Weber polynomials. Finally, we experimentally assess the computational efficiency of the Weber polynomials along with their precision requirements for various discriminant values and compare the results with the theoretical estimates. Our experimental results may be used as a guide for the selection of the most efficient curves in applications residing in resource limited devices such as smart cards that support secure and efficient Public Key Infrastructure (PKI) services.

1 Introduction

Elliptic curve cryptography (ECC) has gained an increasing popularity over the years, as it emerges as a fundamental and efficient technological alternative for building secure public key cryptosystems. This stems from the fact that elliptic curves (ECs) give rise to algebraic structures that offer a number of distinct advantages (smaller key sizes and highest strength per bit) over more customary algebraic structures used in various cryptographic applications (e.g., RSA). The use of smaller parameters for a given level of cryptographic strength results in

^{*} This work was partially supported by the Action IRAKLITOS (Fellowships for Research in the University of Patras) with matching funds from EC and the Greek Ministry of Education.

faster implementations, less storage space, as well as reduced processing and bandwidth requirements. These characteristics make ECC suitable for software as well as for hardware implementations.

The advantage of EC-based cryptography is particularly apparent in applications supporting *Public Key Infrastructure* (PKI) services. In a typical PKI scenario, all users wishing to communicate with each other securely, or access some other services offered by the PKI (e.g., notary services, user authentication, etc), are given two keys, the *public* key and the *private* key. The former is made available to all users through, e.g., a public directory. The latter should be handled with special care in order to avoid accidental disclosure or illicit interception. Protecting the private key is one of the most important issues within a PKI. To this end, the common practice is to generate the two keys within a physically secured (i.e., tamper resistant) device, which is most often a smart card. The public key is exported to a publicly available directory but the private key never leaves the smart card and, thus, it remains safe. It is at this point where elliptic curves can make a difference compared to more conventional schemes such as RSA: since EC-based cryptosystems can achieve the same level of security using much smaller keys and system parameters, the initialization of an EC-based system (generation of the elliptic curve, generation of private and public keys, etc) can be done more economically in hardware as it requires smaller hardware elements (e.g., registers, arithmetic and logic units, etc). Thus, EC-based cryptography seems a very attractive alternative to conventional public key cryptosystems for the support of PKI services. Moreover, in certain PKI applications, a vast number of such ECs may be required to be generated and this should be done as fast as possible. A typical example concerns a wireless and web-based PKI environment in which millions of client devices (e.g., PDAs) connect to secure servers [8]. Clients may be frequently requested to choose different key sizes and EC parameters depending on vendor preferences, security requirements, and processor capabilities. The large number of client connections/transactions along with the (possibly frequent) change of security parameters by the vendor (e.g., due to evolving market conditions and corporate policies) calls for strict timing response constraints not only on the server, but also on the client side.

A frequently employed method for generating elliptic curves with order satisfying certain desirable (security) properties, is the *Complex Multiplication* (CM) method. This method was used by Atkin and Morain in [1] for the construction of elliptic curves with good properties in the context of primality proving while the method was adapted to give rise to curves with good security properties by Spallek [21] and Lay and Zimmer [13] independently. Furthermore, a number of works appeared that compared variants of the CM method and presented experimental results of the construction efficiency such as the recent works of Enge and Morain [5], Müller and Paulus [16] as well as the PhD thesis of Weng [23] and the PhD thesis of Baier [3]. Briefly, the CM method takes as input a given number (usually a prime or a binary power) representing the order of the finite field upon which the EC will be defined and determines a specific parameter,

called the *CM discriminant* D . The EC of the desirable order is generated by constructing certain class field polynomials based on D and finding their roots. The construction and the location of the roots (modulo the finite field's order) of these polynomials is perhaps the most crucial step in the whole process. The most commonly used class field polynomials are the Hilbert (original version of the CM method) and the Weber polynomials. Their main differences are: (i) the coefficients of Hilbert polynomials grow excessively large as the discriminant D increases, while for the same discriminant the Weber polynomials have much smaller coefficients and thus are easier and faster to construct, a thing that is especially desirable in a PKI setting where smart cards are in use; (ii) the roots of the Hilbert polynomial construct directly the EC, while the roots of the Weber polynomial have to be transformed to the roots of its corresponding Hilbert polynomial to construct the EC (a step that is not especially time consuming, however).

The use of Hilbert polynomials in the CM method requires high precision in the arithmetic operations involved in their construction, resulting in considerable increase in computing resources and thus make them not appropriate for fast and frequent generation of ECs within resource limited devices such as smart cards and PDAs. To overcome these shortcomings of Hilbert polynomials, two alternatives have been recently proposed: either to compute them off-line and store them for subsequent use (see, e.g., [18]), or to use Weber polynomials for certain values of D (see, e.g., [2,3,10,12,13,22]) and produce the required Hilbert roots from them. Although the former approach tackles adequately the efficient construction of ECs, there may still be problems with storing and handling several Hilbert polynomials with huge coefficients, especially on cryptographic hardware devices with limited resources. These problems are addressed by the second approach. However, the known studies treat only certain values of D ; for example, the case of $D \equiv 7 \pmod{8}$ and not divisible by 3 is treated in [2,3,10,13], while the cases of $D \not\equiv 3 \pmod{8}$ and $D \not\equiv 0 \pmod{3}$ were treated in [12,22]. To the best of our knowledge, the other cases of D (i.e., $D \equiv 3 \pmod{8}$ and $D \equiv 0 \pmod{3}$) have not been treated before explicitly.

Starting from the fact that it is desirable to work with Weber polynomials in various applications that require the fast and frequent generation of ECs, the goals of this paper are the following: (i) to provide a complete set of transformations of Weber to Hilbert roots that cover all possible values of D , (ii) to prove a theoretical estimate of the precision required for the construction of Weber polynomials and (iii) to provide experiments that indicate values of discriminant D leading to Weber polynomials with small computational requirements and compare their actual precision requirements with the theoretical estimate we give. To the best of our knowledge, no general theoretical treatment exists which gives the required root transformations for *all* possible values of the discriminant D . No general treatment exists either that for each value of D provides an estimation for the precision requirements of the Weber polynomials. We believe that our effort to present an exhaustive list of root transformations and an estimate of the precision requirements of Weber polynomials in a simple, unifying exposi-

tion will be useful to designers of ECC applications, especially applications that will be placed in smart cards in order to support various PKI services.

The rest of the paper is organized as follows. In Section 2 we review some basic definitions and facts about elliptic curves, number theory, and class field polynomials discussing some of their properties relevant to the generation of ECs. In Section 3 we give the transformations that map roots (modulo a prime) of Weber polynomials to roots of Hilbert polynomials. In Section 4 we provide a theoretical estimate of the precision requirements of Weber polynomials and, finally, in Section 5 we give a number of experimental observations that are of relevance to implementations, especially in resource limited devices.

2 Preliminaries

In this section we briefly review the necessary concepts from EC theory, number theory, and the Hilbert and Weber class field polynomials in order to better explain the importance of polynomials in ECC as well as to facilitate the presentation of the root transformations.

2.1 Elliptic Curves

In this work, we consider ECs defined over prime fields. An *elliptic curve* over a (prime) finite field F_p , $p > 3$ and prime, is denoted by $E(F_p)$ and it is comprised of all the points $(x, y) \in F_p$ (in affine coordinates) such that

$$y^2 = x^3 + ax + b, \quad (1)$$

where $a, b \in F_p$ satisfy $4a^3 + 27b^2 \neq 0$. These points, together with a special point denoted by \mathcal{O} (the *point at infinity*) and a properly defined addition operation form an Abelian group. This is the *Elliptic Curve group* and the point \mathcal{O} is its identity element (see [4,20] for more details on this group).

The *order* m of an elliptic curve is the number of the points in $E(F_p)$. The difference between m and p is measured by the so-called *Frobenius trace* $t = p + 1 - m$ for which Hasse's theorem (see e.g., [4,20]) states that $|t| \leq 2\sqrt{p}$, providing upper and lower bounds for m based on p :

$$p + 1 - 2\sqrt{p} \leq m \leq p + 1 + 2\sqrt{p}. \quad (2)$$

The *order of a point* $P \in E(F_p)$ is the smallest positive integer n for which $nP = \mathcal{O}$. Application of Langrange's theorem on $E(F_p)$ dictates that the order of a point $P \in E(F_p)$ divides the order of the elliptic curve group, and hence $mP = \mathcal{O}$ for any $P \in E(F_p)$. This in turn implies that the order of a point cannot exceed the order of the elliptic curve.

Among the most important quantities defined for an elliptic curve $E(F_p)$ given by Eq. (1) are the *curve discriminant* Δ and the *j-invariant*. These two quantities are given by the equations $\Delta = -16(4a^3 + 27b^2)$ and $j = -1728(4a)^3/\Delta$.

For a specific j -invariant $j_0 \in F_p$ (where $j_0 \neq 0, 1728$) two ECs can be readily constructed. This j -invariant is actually a root modulo p of a Hilbert polynomial. Let $k = j_0/(1728 - j_0) \bmod p$. One EC is given by Eq. (1) with $a = 3k \bmod p$ and $b = 2k \bmod p$. The other, which is called the *twist* of the first, is given by

$$y^2 = x^3 + ac^2x + bc^3 \quad (3)$$

with c any quadratic non-residue in F_p . If m_1 and m_2 are the orders of an elliptic curve and its twist respectively, then $m_1 + m_2 = 2p + 2$. This implies that if one of the curves has order $p + 1 - t$, then its twist has order $p + 1 + t$, or vice versa (see [4, Lemma VIII.3]).

2.2 Quadratic Fields and Forms

Let ξ be an algebraic integer (algebraic number satisfying some monic⁵ polynomial equation with integral coefficients), and let f and h be polynomials over \mathbb{Q} . Then, the collection of all numbers of the form $f(\xi)/h(\xi)$, $h(\xi) \neq 0$, constitutes a field denoted by $\mathbb{Q}(\xi)$ and called *the extension of \mathbb{Q} by ξ* . If ξ is a root of an irreducible¹ quadratic polynomial over \mathbb{Q} , then $\mathbb{Q}(\xi)$ is called a *quadratic field*. Additional information on algebraic numbers and quadratic fields can be found in [17].

Let D be a positive integer which is not divisible by any square of an odd prime and which satisfies $D \equiv 3 \pmod{4}$ or $D \equiv 4, 8 \pmod{16}$. The quantity $-D < 0$ is called a *fundamental discriminant*. The subset of integers in $\mathbb{Q}(\sqrt{-D})$ forms a ring which is denoted by \mathbb{O} . A *quadratic form* of a discriminant $-D$ is a 3-tuple of integers $[a, b, c]$ such that $b^2 - 4ac = -D$. The form is called *primitive* if $\gcd(a, b, c) = 1$, and *reduced* if $|b| \leq a \leq c$ and $b \geq 0$ whenever $c = a$ or $|b| = a$.

There is a natural correspondence between a quadratic form $[a, b, c]$ and the root τ of the quadratic equation $az^2 + bz + c = 0$ with $\text{Im}(\tau) > 0$: $-D$ is the discriminant of τ and $\tau = (-b + \sqrt{-D})/2a$. It can be proved that the set of primitive reduced quadratic forms of discriminant $-D$, denoted by $\mathcal{H}(-D)$, is finite. Moreover, it is possible to define an operation that gives to $\mathcal{H}(-D)$ the structure of an Abelian group whose neutral element is called the *principal form*. The order of $\mathcal{H}(-D)$ is denoted by $h(-D)$, or simply h if $-D$ is clear from the context. The principal form is equal to $[1, 0, D/4]$ if $D \equiv 0 \pmod{4}$, and to $[1, -1, (D+1)/4]$ if $D \equiv 3 \pmod{4}$. In this case, the root of the quadratic equation, denoted by τ^* , is $\tau^* = \sqrt{-D}/2$ if $D \equiv 0 \pmod{4}$, and $\tau^* = (1 + \sqrt{-D})/2$ if $D \equiv 3 \pmod{4}$.

2.3 Class Field Polynomials

Let τ_k be the root corresponding to a reduced positive primitive quadratic form $[a_k, b_k, c_k] \in \mathcal{H}(-D)$. Then the *class equation* of $\mathbb{O} \subset \mathbb{Q}(\sqrt{-D})$, or *Hilbert polynomial*, is defined by

⁵ A polynomial is called monic if its leading coefficient is 1, and irreducible if it cannot be written as a product of two polynomials.

$$H_D(x) = \prod_{k=1}^h (x - j(\tau_k)) \quad (4)$$

where the quantity $j(\tau_k)$, for $\tau_k \in \mathbb{O}$, is called a *class invariant* of \mathbb{O} . In particular, for any $\tau \in \mathbb{O}$, $j(\tau)$ is defined as

$$j(\tau) = \frac{(256\theta(\tau) + 1)^3}{\theta(\tau)},$$

where $z = e^{2\pi\sqrt{-1}\tau}$, $\theta(\tau) = \frac{\Delta(2\tau)}{\Delta(\tau)}$, and

$$\Delta(\tau) = z \left(1 + \sum_{n \geq 1} (-1)^n \left(z^{n(3n-1)/2} + z^{n(3n+1)/2} \right) \right)^{24}.$$

The class invariants $j(\tau_k)$ (which are the roots of $H_D(x)$) generate a field over $\mathbb{Q}(\sqrt{-D})$ called the *Hilbert class field*.

Alternative generators of the class field can be provided by singular values of other functions. Such functions are powers of the Weber functions which generate the Weber polynomials.

The first main theorem of complex multiplication says that the class invariant $j(\tau)$, $\tau \in \mathbb{O}$, generates the Hilbert class field over $\mathbb{Q}(\sqrt{-D})$. Weber considered the explicit construction of the Hilbert class field using other modular functions $g(z)$. When $g(\tau)$ and $j(\tau)$ generate the same field over $\mathbb{Q}(\sqrt{-D})$, $g(\tau)$ is also called a *class invariant* of \mathbb{O} , and its minimal polynomial $W_D(x)$ is called the *reduced class equation* or the *Weber polynomial*. Weber polynomials are defined using the Weber functions (see [1,9])

$$\begin{aligned} f(z) &= q^{-1/48} \prod_{m=1}^{\infty} (1 + q^{(m-1)/2}) & f_1(z) &= q^{-1/48} \prod_{m=1}^{\infty} (1 - q^{(m-1)/2}) \\ f_2(z) &= \sqrt{2} \, q^{1/24} \prod_{m=1}^{\infty} (1 + q^m) & \text{where } q &= e^{2\pi z \sqrt{-1}}. \end{aligned}$$

Then, the Weber polynomial $W_D(x)$ is defined as

$$W_D(x) = \prod_{\ell=1}^{h'} (x - g(\tau_\ell)) \quad (5)$$

where $g(\tau_\ell)$ (a class invariant of $W_D(x)$) is an expression – depending on the value of D – of the Weber functions, $\tau_\ell \in \mathbb{O}$ satisfies the equation $a_\ell z^2 + 2b_\ell z + c_\ell = 0$ for which $4b_\ell^2 - 4a_\ell c_\ell = -4d$, where $d = D/4$ if $D \equiv 0 \pmod{4}$, and $d = D$ if $D \equiv 3 \pmod{4}$, and h' (the degree of $W_D(x)$) can be either h or $3h$. All class invariants for the different values of D will be defined in Section 3.

We would like to mention that the possible class invariants for a given discriminant D are potentially infinite, giving rise to different class polynomials

and consequently to the problem of which one to use (for details see [5,6,19]). A comparison of all possible class invariants for a given D was made in [5] using as criterion the *height*⁶ of their minimal polynomials, since it is computationally easier to use invariants that produce polynomials of small height. In particular, it is shown in [5] that Weber polynomials is one of the best choices between all possible polynomials. In addition, an ordering of class invariants (from the best to the worst) is made in [5], where the best invariant is the one with minimum height. According to this ordering, the cases of $D \equiv 0 \pmod{3}$ are “worse” than the cases of $D \not\equiv 0 \pmod{3}$.

3 Transformations of Weber Roots to Hilbert Roots

In this section, we will describe a complete set of transformations of the roots of the Weber polynomials to the roots of the corresponding (generated by the same D) Hilbert polynomials. The transformations will be given for all possible values of discriminant D .

In order to explain the transformations that will follow, we need two relations. From the definition of the Weber functions and the class invariant $j(\tau)$ that generates the Hilbert polynomials, one can readily obtain the following relations:

$$f(\tau)f_2\left(\frac{1+\tau}{2}\right) = e^{\pi\sqrt{-1}/24}\sqrt{2} \quad (6)$$

$$j(\tau) = \frac{(A-16)^3}{A}, \quad (7)$$

where $A \in \{f^{24}(\tau), -f_1^{24}(\tau), -f_2^{24}(\tau)\}$.

Recall from Section 2.3 that the Weber polynomial $W_D(x)$ is generated by its class invariants which are also the roots of the polynomial. The real roots of $W_D(x)$, for all values of D , are given by the class invariants presented in Tables 1 and 2. There are ten cases of discriminant D that define ten different class invariants. Recall from Section 2.2 that D is either $3 \pmod{4}$ or $4, 8 \pmod{16}$ and that $d = D/4$ if $D \equiv 0 \pmod{4}$, and $d = D$ if $D \equiv 3 \pmod{4}$. This in turn implies that $d \equiv 3, 7 \pmod{8}$ if $D \equiv 3 \pmod{4}$, while $d \equiv 1, 2, 5, 6 \pmod{8}$ when $D \equiv 4, 8 \pmod{16}$. The ten class invariants split into two groups of five each, depending on whether $D \not\equiv 0 \pmod{3}$ or $D \equiv 0 \pmod{3}$. Tables 1 and 2 give the details.

We verified the correctness of the class invariants given in Tables 1 and 2 using the IEEE Standard P1363 [9]. We would like to mention that: (i) the above class invariants have been very recently given in [15], where, however a different invariant is presented for the case of $d \equiv 3 \pmod{8}$ in Table 2, which does not agree with the IEEE Standard P1363 [9]; (ii) the class invariants of Table 1 are also given in [1,11,22], where however a different invariant for the case of $d \equiv 5 \pmod{8}$ is presented which does not agree with both the IEEE

⁶ The logarithm of the largest coefficient of the polynomial.

$d \bmod 8$	class invariant
1	$f^2(\sqrt{-d})/\sqrt{2}$
2 or 6	$f_1^2(\sqrt{-d})/\sqrt{2}$
3	$f(\sqrt{-d})$
5	$f^4(\sqrt{-d})/2$
7	$f(\sqrt{-d})/\sqrt{2}$

Table 1. Class invariants for $D \not\equiv 0 \pmod{3}$

$d \bmod 8$	class invariant
1	$f^6(\sqrt{-d})/(2\sqrt{2})$
2 or 6	$f_1^6(\sqrt{-d})/(2\sqrt{2})$
3	$f^3(\sqrt{-d})/2$
5	$f^{12}(\sqrt{-d})/2^3$
7	$f^3(\sqrt{-d})/(2\sqrt{2})$

Table 2. Class invariants for $D \equiv 0 \pmod{3}$

Standard P1363 [9] and the class invariant for the same case ($D \not\equiv 0 \pmod{3}$) given in [15].

In both applications that we described, it is necessary to obtain the roots modulo a prime p of the Hilbert polynomial and we want to achieve this by using the roots (modulo p) of Weber polynomials. Hence, there must be a way to retrieve a root modulo p of the Hilbert polynomial $H_D(x)$ from a root modulo p of the corresponding Weber polynomial $W_D(x)$. It can be shown that if we can find a transformation $T(\cdot)$ of a real root $g(\tau_\ell)$ of the Weber polynomial to a real root $j(\tau_i)$ of the corresponding Hilbert polynomial, then the same transformation will hold for the roots of the polynomials modulo p .

We have already mentioned that the class invariants given in Tables 1 and 2 represent the real roots of the Weber polynomials. We shall refer to all these values by $F(\sqrt{-d})$, where F depends on the value of D . It is also known that when τ^* corresponds to a principal form, then the class invariant $j(\tau^*)$ of $H_D(x)$ is a real root. Hence the goal is to find a transformation $T(\cdot)$ such that $j(\tau^*) = T(F(\sqrt{-d}))$.

The idea is as follows. From Equation (7) we know that if one of $f^{24}(\tau^*)$, $-f_1^{24}(\tau^*)$, $-f_2^{24}(\tau^*)$ can be calculated, then $j(\tau^*)$ can also be computed. The problem now is reduced in finding one of $f^{24}(\tau^*)$, $-f_1^{24}(\tau^*)$, $-f_2^{24}(\tau^*)$ from $F(\sqrt{-d})$. When $D \equiv 0 \pmod{4}$, then $\tau^* = \sqrt{-d}$, and finding $f^{24}(\sqrt{-d})$, or $-f_1^{24}(\sqrt{-d})$, or $-f_2^{24}(\sqrt{-d})$ from $F(\sqrt{-d})$ is rather easy. When $D \equiv 3 \pmod{4}$ however, then $\tau^* = \frac{1+\sqrt{-d}}{2}$, and finding $f^{24}(\frac{1+\sqrt{-d}}{2})$, or $-f_1^{24}(\frac{1+\sqrt{-d}}{2})$, or $-f_2^{24}(\frac{1+\sqrt{-d}}{2})$ from $F(\sqrt{-d})$ is a little more complicated (actually, we have to use Equation (6) for these cases).

In the following, we present the details for three different cases of D of the transformation of a real root R_W of a Weber polynomial to a real root R_H of the corresponding Hilbert polynomial. The remaining seven cases can be derived with a similar way and are not described due to lack of space.

(a) $D \equiv 7 \pmod{8}$ and $D \not\equiv 0 \pmod{3}$: From Table 1, the class invariant in this case is $\frac{f(\sqrt{-d})}{\sqrt{2}} = R_W$. Since $d \equiv 3 \pmod{4}$, we have that $\tau^* = (1 + \sqrt{-d})/2$. Using Equation (6), we get

$$f_2(\tau^*) = f_2\left(\frac{1 + \sqrt{-d}}{2}\right) = e^{\frac{\pi\sqrt{-1}}{24}} \sqrt{2} f^{-1}(\sqrt{-d}) \Rightarrow f_2(\tau^*) = e^{\frac{\pi\sqrt{-1}}{24}} R_W^{-1}$$

$$\Rightarrow -f_2^{24}(\tau^*) = R_W^{-24} = A.$$

Thus, from Equation (7) we obtain

$$R_H = \frac{(A - 16)^3}{A} = \frac{(R_W^{-24} - 16)^3}{R_W^{-24}}.$$

(b) $D \equiv 3 \pmod{8}$ and $D \not\equiv 0 \pmod{3}$: This is a very interesting case because the degree of the Weber polynomial is three times larger than the degree of the corresponding Hilbert polynomial. This case is frequently referred to as problematic, because of the high degree of the resulting Weber polynomials. If the number of distinct roots of the Weber polynomial is exactly three times the number of the roots of the corresponding Hilbert polynomial, the transformation below maps three roots of the Weber polynomial into the same root of the Hilbert polynomial. Obviously, when the number of distinct roots of the Weber polynomial is equal to the number of distinct roots of the Hilbert polynomial, one root of the Weber polynomial is mapped to exactly one root of the Hilbert polynomial.

Since $d = D \equiv 3 \pmod{8}$, we have that $\tau^* = \frac{1 + \sqrt{-d}}{2}$, and from Table 1 the class invariant is $f(\sqrt{-d}) = R_W$. According to Equation (6) we have

$$f_2(\tau^*) = f_2\left(\frac{1 + \sqrt{-d}}{2}\right) = e^{\frac{\pi\sqrt{-1}}{24}} \sqrt{2} f^{-1}(\sqrt{-d}) \Rightarrow -f_2^{24}(\tau^*) = 2^{12} R_W^{-24} = A.$$

Hence, by Equation (7)

$$R_H = \frac{(A - 16)^3}{A} = \frac{(2^{12} R_W^{-24} - 16)^3}{2^{12} R_W^{-24}}.$$

(c) $D/4 \equiv 2, 6 \pmod{8}$ and $D \not\equiv 0 \pmod{3}$: For this value of $D = 4d$, we have that $\tau^* = \sqrt{-d}$, and the class invariant is $f_1^2(\sqrt{-d})/\sqrt{2} = R_W$ (see Table 1). Then,

$$f_1^2(\tau^*) = f_1^2(\sqrt{-d}) = \sqrt{2} R_W \Rightarrow -f_1^{24}(\tau^*) = -2^6 R_W^{12} = A$$

which by Equation (7) gives

$$R_H = \frac{(A - 16)^3}{A} = \frac{(-2^6 R_W^{12} - 16)^3}{-2^6 R_W^{12}} = \frac{(2^6 R_W^{12} + 16)^3}{2^6 R_W^{12}}.$$

Finally, we would like to remark that in the IEEE Standard P1363 [9] formulas are given that lead directly from the roots of Weber polynomials to the parameters of the ECs, without involving transformations to roots of Hilbert polynomials. However, we believe that the transformations given in our paper, not only are they not slower than the formulas given in the Standard but they also give a justification as to how the roots of Weber polynomials lead to the construction of ECs.

4 Precision Requirements of Weber Polynomials

In this section we will focus on the precision required for the construction of the Weber polynomials. First, for reasons of comparison and completeness, we note that a very accurate estimation of the bit precision of Hilbert polynomials made in [13] gives an upper bound of $3.32(\Lambda_H + h/4 + 5)$, where $\Lambda_H = \frac{\pi\sqrt{D}}{\ln 10} \sum_{k=1}^h \frac{1}{a_k}$. Our experiments showed that this bound is remarkably accurate.

Let $\Lambda_W = \frac{\pi\sqrt{D}}{\ln 2} \sum_{\ell} \frac{1}{a_{\ell}}$, where the sum runs over the same values of ℓ as the product in Eq. (5). The bit precision required for the construction of the Weber polynomial is upper bounded by $v_0 + \Lambda_W$ (see e.g., [22]), where v_0 is a positive constant that handles round-off errors (typically $v_0 = 33$). This estimate of precision can be, however, much larger than the actual precision required for the Weber polynomials. For the case of $D \equiv 7 \pmod{8}$ and not divisible by 3, a better upper bound of $3.32(1 + (\Lambda_H + h/4 + 5)/47)$ is provided by [13]. However, this precision estimate can not be used in other cases of D . For this reason, we provide in the following lemma a new precision estimate that covers all values of discriminant D .

Lemma 1. *The bit precision required for the construction of Weber polynomials for various values of the discriminant D is approximately*

$$c_1 h + \frac{\pi\sqrt{D}}{c_2 \ln 2} \sum_{\ell} \frac{1}{a_{\ell}},$$

where the sum runs over the same values of ℓ as the product in Eq. (5) and the constants c_1 and c_2 are given by

$$c_1 = \begin{cases} 3 & \text{if } D \equiv 3 \pmod{8} \\ 1 & \text{if } D \not\equiv 3 \pmod{8} \end{cases} \quad (8)$$

$$c_2 = \begin{cases} 24 & \text{if } D \equiv 3, 7 \pmod{8} \text{ and } D \not\equiv 0 \pmod{3} \\ 8 & \text{if } D \equiv 3, 7 \pmod{8} \text{ and } D \equiv 0 \pmod{3} \\ 6 & \text{if } D/4 \equiv 5 \pmod{8} \text{ and } D \not\equiv 0 \pmod{3} \\ 2 & \text{if } D/4 \equiv 5 \pmod{8} \text{ and } D \equiv 0 \pmod{3} \\ 12 & \text{if } D/4 \equiv 1, 2, 6 \pmod{8} \text{ and } D \not\equiv 0 \pmod{3} \\ 4 & \text{if } D/4 \equiv 1, 2, 6 \pmod{8} \text{ and } D \equiv 0 \pmod{3} \end{cases} \quad (9)$$

Proof. For the case $D \not\equiv 3 \pmod{8}$ and from the proof of Proposition (B4.4) in [11], if the Weber polynomial is written in the form $W_D(x) = x^h + w_{h-1}x^{h-1} + \dots + w_1x + w_0$ then $|w_i| \leq 2^h M$, where $M = \prod_{\ell} \max(1, |g(\tau_{\ell})|)$. This means that the bit precision required for the coefficient w_i of the polynomial is $\log_2(|w_i|) \leq h + \log_2 M \leq h + \sum_{\ell} \log_2(|g(\tau_{\ell})|)$. Therefore, the bit precision required for the construction of the whole polynomial (i.e., the construction of its coefficients) is at most $h + \sum_{\ell} \log_2(|g(\tau_{\ell})|)$ and thus $c_1 = 1$. If $D \equiv 3 \pmod{8}$, then the degree of $W_D(x)$ is equal to $3h$ and repeating the same steps as above, we conclude

that the bit precision is at most $3h + \sum_{\ell} \log_2(|g(\tau_{\ell})|)$ which gives $c_1 = 3$. Thus, we need to estimate the precision requirements for the computation of $g(\tau_{\ell})$.

The precision required by each $g(\tau_{\ell})$ is related to the precision required by $f(\tau_{\ell})$, $f_1(\tau_{\ell})$ or $f_2(\tau_{\ell})$ as evidenced by Table 1 and 2. We observe that, in general, g is equal to one of these functions raised to a constant power K and multiplied by a small constant which we will ignore in the following computations as it will affect the final estimate only by a very small additive constant. This implies that the precision needed for $g(\tau_{\ell})$ is approximately K times the precision needed for $f(\tau_{\ell})$, $f_1(\tau_{\ell})$ or $f_2(\tau_{\ell})$.

In addition, it is known that $j(z) = \frac{(f^{24}(z)-16)^3}{f^{24}(z)} = \frac{(f_1^{24}(z)+16)^3}{f_1^{24}(z)} = \frac{(f_2^{24}(z)+16)^3}{f_2^{24}(z)}$. These equalities imply that the precision needed for $j(\tau_{\ell})$ is approximately 48 times the precision needed for $f(\tau_{\ell})$, $f_1(\tau_{\ell})$ or $f_2(\tau_{\ell})$. Using the expansion of j in terms of its Fourier series (see [4]), we obtain that $|j(\tau_{\ell})| \approx |e^{-2\pi\sqrt{-1}\ell}| = e^{2\pi\sqrt{D}/\alpha}$. Therefore, the bit precision that is required for the computation of $j(\tau_{\ell})$ is $\log_2 |j(\tau_{\ell})| \approx \frac{2\pi\sqrt{D}}{\alpha \ln 2}$ and, consequently, the precision required for $g(\tau_{\ell})$ is given by $\log_2 |g(\tau_{\ell})| \approx \frac{K}{48} \log_2 |j(\tau_{\ell})| = \frac{2K\pi\sqrt{D}}{48\alpha \ln 2} = \frac{K\pi\sqrt{D}}{24\alpha \ln 2}$. This, in turn, results in the total bit precision requirements for the computation of the Weber polynomial: $c_1 h + \frac{K\pi\sqrt{D}}{24\alpha \ln 2} \sum_{\ell} \frac{1}{\alpha_{\ell}}$.

We will show for a case of D how we can derive the constant c_2 : for the case $D \equiv 3 \pmod{8}$ and $D \equiv 0 \pmod{3}$, the precision required by $g(\tau_{\ell})$ is approximately three times (i.e., $K = 3$) the precision required by $f(\tau_{\ell})$, $f_1(\tau_{\ell})$ or $f_2(\tau_{\ell})$ as it is evident from Table (2). Thus we obtain that the bit precision required in this case is given by $3h + \frac{3\pi\sqrt{D}}{24\alpha \ln 2} \sum_{\ell} \frac{1}{\alpha_{\ell}}$. This results to $c_2 = 8$. Using the same reasoning we can compute the bit precision requirements for the other cases of D , completing the proof of the lemma. \square

We would like to point out that our lemma suggests an ordering (based on the theoretical estimates) of the bit precision requirements for the different values of discriminant D . For example, it seems that the case $D \equiv 3, 7 \pmod{8}$ and $D \not\equiv 0 \pmod{3}$ requires less precision than the other cases and this can mean that these values of discriminants are better for implementations. This ordering is verified by our experiments described in Section 5 below.

5 Experimental Results

In this section, we present an experimental study regarding the computational efficiency and the bit precision requirements of Weber polynomials and how these requirements are compared with their approximated precision requirements using Lemma 1. Our implementations for generating the polynomials are actually part of a variant of the CM method that generates ECs of a desirable order and uses both Weber and Hilbert polynomials. In contrast with other implementations (see e.g., [2,3]) which have been carried out in C++ and use advanced C++ libraries (e.g., LiDIA [14]) our implementations have been carried out in ANSI C using the (ANSI C) library GNUMP [7] (for high precision floating

point arithmetic and also for the generation and manipulation of integers of unlimited precision), for maximum portability. Our implementation and experimentation environment was a Pentium III PC (933 MHz) running Linux 2.4.10 and equipped with 256 MB of main memory. The code for the generation of ECs using Weber polynomials had size 53KB, while the code for the generation of ECs using Hilbert polynomials had size 49KB.

The construction of Hilbert and Weber polynomials require high-precision complex and floating point arithmetic with the greater demands placed by Hilbert polynomials. Also, their construction required the implementation of functions such as $\cos(x)$, $\sin(x)$, $\exp(x)$, $\ln(x)$, $\arctan(x)$ and \sqrt{x} , which were implemented using their Taylor series expansion. Since the basic complex number algebraic operations (addition, multiplication, exponentiation, and squaring) as well as a high precision floating point implementation of the other involved functions did not exist in GNUMP, we had to implement them from scratch.

In our experiments we have mainly focused on measuring the time (in secs) for the construction of Weber polynomials, the bit precision they require (number of bits that are required by the GNUMP library), and the difference between the actual precision and the approximated precision from Lemma 1. In Figure 1 we present the approximated vs. the actual bit precision for Weber polynomials with even discriminant D . The degree h of the polynomials ranges from 4 to 52 as the discriminant D increases from 228 to 9924. We observe that all the estimates of the precision are larger than the actual precision. However, for values of discriminant D that are not divided by 3 the estimation of the precision is close to the actual one, while for the other cases the estimation can be much larger. However, for all the cases of D , the estimate of precision is very close to the actual one for polynomials with small h ($h < 20$, corresponding to $D < 2500$ approximately). Similar observations hold for odd values of the discriminant D .

Figure 1 indicates also that there is an ordering in the precision figures for various values of D . It can be seen, for instance, that the case of $D/4 \equiv 1, 2, 6 \pmod{8}$ and $D \not\equiv 0 \pmod{3}$ requires less precision than the case $D/4 \equiv 5 \pmod{8}$ and $D \not\equiv 0 \pmod{3}$, which in turn is better than the case $D/4 \equiv 1, 2, 6 \pmod{8}$ and $D \equiv 0 \pmod{3}$. The worst case is $D/4 \equiv 5 \pmod{8}$ and $D \equiv 0 \pmod{3}$. The case of D that requires the least precision among all the possible values is $D \equiv 3, 7 \pmod{8}$ and $D \not\equiv 0 \pmod{3}$ (it is not included in Figure 1, because in this figure we report only on even values of D). Note also that a similar ordering is implied by the estimates provided by Lemma 1.

The difference in the precision requirements for the various values of discriminant D is reflected in the time requirements for the construction of the polynomials. In Figure 2, we summarize all possible cases of D . The degree h of the polynomials ranges from 50 to 150 and D from 10766 to 69396. It is evident that there is, again an ordering among the different cases of the discriminant. The worst case is $D/4 \equiv 5 \pmod{8} \equiv 0 \pmod{3}$, while the best is $D \equiv 3, 7 \pmod{8} \not\equiv 0 \pmod{3}$ (for $D = 68383$ and $h = 148$ the time for the construction of the polynomial is only 4.43 seconds). We observe that the three worst cases are those that correspond to values of D that are divided by 3. The ordering

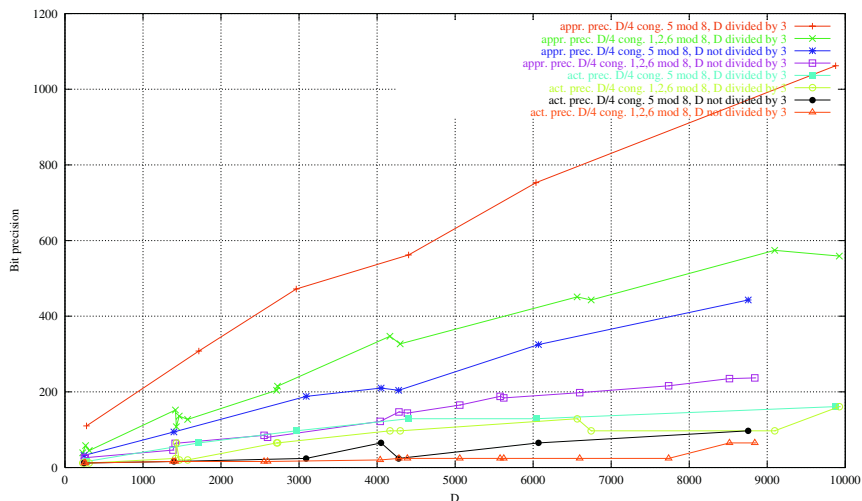


Fig. 1. Actual and approximated bit precision for the construction of Weber polynomials

between the two groups of D (divided or not divided by 3) is the same: the worst case is $D/4 \equiv 5 \pmod{8}$, then is the case of $D/4 \equiv 1, 2, 6 \pmod{8}$ and the best one is $D \equiv 3, 7 \pmod{8}$. We also note that the same ordering is observed in the precision requirements for the various values of D . This apparent ordering may be helpful to the designers of ECCs as it may be used as a guideline for the selection of values of D that lead to Weber polynomials with the least computational requirements.

6 Conclusions

In this paper we have considered the use of Weber polynomials in elliptic curve cryptography implemented in resource limited devices, such as smart cards and PDAs. Weber polynomials are important in EC-based cryptography since their coefficients are considerably smaller than the coefficients of the corresponding Hilbert polynomials and can, thus, be manipulated with ease by resource limited devices that are used within PKIs. The only problem is that the roots of the Weber polynomials do not lead directly to the construction of the EC parameters and should be, first, transformed into equivalent Hilbert polynomial roots. To this end we have presented in this paper, in a unifying and simple manner, all the transformation of roots of Weber polynomials into roots of the corresponding Hilbert polynomials as well as estimates for the precision requirements of Weber polynomials for all possible discriminant values. We have also conducted experiments and compared the construction efficiency of Weber polynomials for various discriminant values. We observed that there is an ordering among the values of D that is defined by its divisibility properties. We believe that our

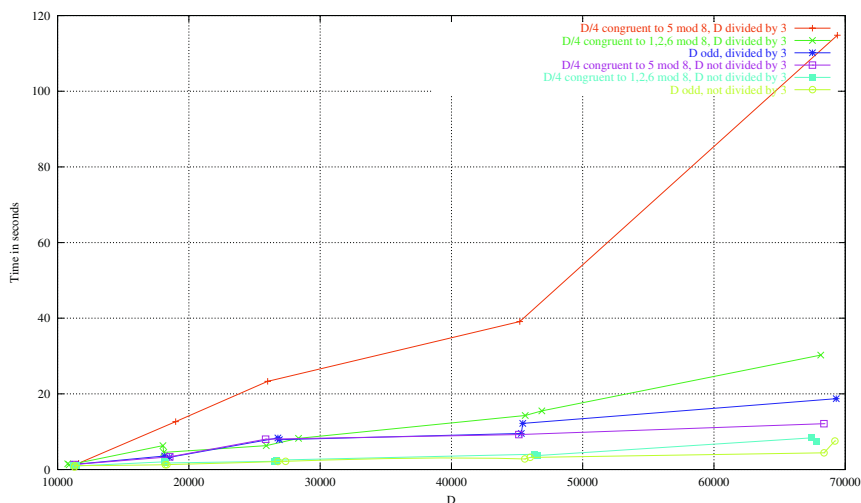


Fig. 2. Time for the construction of Weber polynomials

experimental results can be used as a guideline for the selection of the appropriate class field polynomials when constructing elliptic curves as the potential designer can have an estimate of the computation time as well as the precision required before the actual implementation is accomplished within some devices with limited computing power.

References

1. A.O.L. Atkin and F. Morain, Elliptic curves and primality proving, *Mathematics of Computation* **61**, pp. 29–67, 1993.
2. H. Baier and J. Buchmann, Efficient construction of cryptographically strong elliptic curves, in *Progress in Cryptology – INDOCRYPT 2000*, Lecture Notes in Computer Science Vol. 1977, Springer-Verlag, pp. 191–202, 2000.
3. H. Baier, Efficient Algorithms for Generating Elliptic Curves over Finite Fields Suitable for Use in Cryptography, PhD Thesis, Dept. of Computer Science, Technical Univ. of Darmstadt, May 2002.
4. I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography*, London Mathematical Society Lecture Note Series 265, Cambridge University Press, 1999.
5. A. Enge and F. Morain, Comparing invariants for class fields of imaginary quadratic fields, in *Algorithmic Number Theory – ANTS-V*, Lecture Notes in Computer Science Vol. 2369, Springer-Verlag, pp. 252–266, 2002.
6. A. Enge and R. Schertz, *Constructing Elliptic Curves from Modular Curves of Positive Genus*, Preprint, March 2003.
7. GNU multiple precision library, edition 3.1.1, September 2000. Available at: <http://www.swox.com/gmp>.
8. N. Gura, H. Eberle, and S.C. Shantz, Generic Implementations of Elliptic Curve Cryptography using Partial Reduction, in *Proc. 9th ACM Conf. on Computer and Communications Security – CCS’02*, pp.108–116.

9. IEEE P1363/D13, *Standard Specifications for Public-Key Cryptography*, ballot draft, 1999. <http://grouper.ieee.org/groups/1363/tradPK/draft.html>.
10. E. Kaltofen, T. Valente, and N. Yui, An Improved Las Vegas Primality Test, in *Proc. ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, pp. 26-33, 1989.
11. E. Kaltofen and N. Yui, Explicit construction of the Hilbert class fields of imaginary quadratic fields by integer lattice reduction. Research Report 89-13, Rensselaer Polytechnic Institute, May 1989.
12. E. Konstantinou, Y.C. Stamatiou, and C. Zaroliagis, On the Efficient Generation of Elliptic Curves over Prime Fields, in *Cryptographic Hardware and Embedded Systems – CHES 2002*, Lecture Notes in Computer Science Vol. 2523, Springer-Verlag, pp. 333–348, 2002.
13. G.J. Lay and H. Zimmer, Constructing Elliptic Curves with Given Group Order over Large Finite Fields, in *Algorithmic Number Theory – ANTS-I*, Lecture Notes in Computer Science Vol. 877, Springer-Verlag, pp.250-263, 1994.
14. LiDIA. *A library for computational number theory*, Technical University of Darmstadt. Available from <http://www.informatik.tu-darmstadt.de/TI/LiDIA/Welcome.html>.
15. F. Morain, Computing the cardinality of CM elliptic curves using torsion points, Preprint, October 2002.
16. V. Müller and S. Paulus, On the Generation of Cryptographically Strong Elliptic Curves, preprint, 1997.
17. I. Niven, H.S. Zuckerman, and H.L. Montgomery, *An Introduction to the Theory of Numbers*, John Wiley & Sons, 5th edition, 1991.
18. E. Savaş, T.A. Schmidt, and Ç.K. Koç, Generating Elliptic Curves of Prime Order, in *Cryptographic Hardware and Embedded Systems – CHES 2001*, LNCS Vol. 2162 (Springer-Verlag, 2001), pp. 145-161.
19. R. Schertz, Weber's class invariants revisited, *J. Théor. Nombres Bordeaux* **14:1**, 2002.
20. J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, GTM 106, 1986.
21. A.-M. Spallek, *Konstruktion einer elliptischen Kurve über einem endlichen Körper zu gegebener Punktgruppe*, Master Thesis, Universität GH Essen, 1992.
22. T. Valente, *A distributed approach to proving large numbers prime*, Rensselaer Polytechnic Institute Troy, New York, PhD Thesis, August 1992.
23. A. Weng, *Konstruktion kryptographisch geeigneter Kurven mit komplexer Multiplikation*, PhD thesis, Institut für Experimentelle Mathematik, Universität GH Essen, 2001.

Threshold Password-Based Authentication Using Bilinear Pairings^{*}

Songwon Lee¹, Kyusuk Han¹, Seok-kyu Kang¹, Kwangjo Kim¹, and
So Ran Ine²

¹ Information and Communications University (ICU)

119, Munjiro, Yuseong-Gu, Daejeon, Korea

{swonlee, hankyusuk, redorb, kkj}@icu.ac.kr

² NITZ. Corp.

San 14-1 Koejong-dong, Seo-gu, Daejeon, Korea

srine@nitz.co.kr

Abstract. We present a new threshold password-based authentication protocol that allows a roaming user(a user who accesses a network from different client terminals) to download a private key from remote servers with knowledge of only his identity and password. He does not need to carry the smart card storing his private information. We aim that a protocol has to allow a user to get his private key from the servers, even if some of the servers are compromised under the multi-server roaming system. In this paper, we firstly suggest a threshold password-only roaming protocol using (k,n) -threshold scheme which only k honest servers or more are engaged to reconstruct a secret key. Our scheme is based on bilinear pairings which could be built from Weil pairing or Tate pairing.

1 Introduction

With rapid development on the Internet a user Bob can easily access the network to get some services from a service provider, or to retrieve his sensitive private data stored in the server previously. In that case, he has to convince the server that he is a really legitimate user. To verify the identity(ID for short) of a user many real systems use password-based authentication. The fundamental problems with passwords come from the fact that most users' passwords are drawn from a relatively small spaces and are easily memorable, which also means that the password may be easily guessed by an attacker.

Let us assume that a roaming user accesses a network from different client terminals to download a private key from remote servers with knowledge of only his ID and password. He does not need to carry the smart card storing his private information. While the smart card plays an important role in storing sensitive information, it is impractical in many real environments due to inconvenience, for example, a user needs an external interface device to communicate with a

^{*} This work was partially supported by R&D Program for Fusion Strategy of Advanced Technologies, Ministry of Science and Technology in Korea

smart card. Focused on this point, strong password authentication protocols were presented by Perlman *et al.*[16], Ford *et al.*[6], Jablon[9], *etc.* Some of them used multiple servers to implement a roaming protocol that uses a weak secret key to securely retrieve and reconstruct a strong private key that has been divided into pieces distributed among multiple servers. We note that as one of goals, a protocol has to allow a user to get his private key from the servers, even if some of the servers are compromised.

In this paper we present a threshold password-based authentication protocol for a roaming user. We use a (k,n) -threshold scheme in which only k honest servers or more are engaged to reconstruct a secret key. Our scheme is based on bilinear pairings that could be built from Weil pairing or Tate pairing over *Gap Diffie-Hellman*(GDH) group, which *Computational Diffie-Hellman*(CDH) problem is hard but *Decision Diffie-Hellman*(DDH) problem is easy.

2 Preliminaries

2.1 Previous Works

Perlman and Kaufman [16] presented protocols that we can securely retrieve a private key and use this to download the rest of our security context. Ford and Kaliski [6] proposed methods that use multiple servers to prevent attacking by introducing *password hardening protocol* by which servers interact with the user to harden the user's password into a strong secret without revealing either the user's password or the hardened result. A *password-only multi-server roaming protocol* is presented by Jablon [9]. In his protocol, the user can authenticate servers and retrieve his private key for temporary use on a client machine using just an easily memorable password. [6] and [9] make use of the multiple servers to gain the goal of the protocol. When some of the servers are compromised, the user can not obtain valid secret key no matter what the user has a method to verify the key. In our proposed scheme, we deal with this problem.

Let's briefly review here the protocol proposed in [9]. In this protocol the author introduced *forgiveness protocol* by which user's honest mistakes are forgiven by sending evidence of recent prior invalid access attempts after each successful authentication. But we do not consider here this forgiveness.

Parameters. The protocol operates in a subgroup of order q in \mathbb{Z}_p^* , where p, q and r are odd primes, $p = 2rq + 1$, $2^{k-1} < p < 2^k$, $r \neq q$, and $2^{2j-1} < q < 2^{2j}$.

Enrollment. The user, Alice, selects a password π , computes $g_\pi = h(\pi)^{2r}$, and creates a private key U . For each $i \in [1, n]$, she computes a secret key share $S_i = g_\pi^{y_i}$ using randomly chosen $y_i \in_{\mathcal{R}} [1, q-1]$. She then generates her master j -bit symmetric key with $K_m = h(S_1 \parallel \dots \parallel S_n) \bmod 2^j$, her encrypted private key as $U_K =_{K_m} \{U\}$, and her key verifier $proof_{PK_m} = h(K_m \parallel g)$.

1. *Client*: send $\{ID_A, y_i, U_K, proof_{PK_m}\}$ to each server L_i for all $i \in [1, n]$.
2. *Servers*: store them in a list C_i maintained on each server.

Authenticated Retrieval. To retrieve her master key at a later time, the client and servers perform the protocol as below:

1. *Client*: select a random number $x \in [1, q - 1]$, computes $X = g^{\pi^x} \bmod p$, and then send $\{ID_A, X\}$ to *Servers*.
2. *Servers*: retrieve $\{ID_A, y_i, U_K, proof_{PK_m}\}$ from C_i , compute $Y_i = X^{y_i}$, and then reply $\{Y_i, U_K, proof_{PK_m}\}$ to *Client*.
3. *Client*: compute $S_i = Y_i^{1/x} \bmod p$ for each $i \in [1, n]$, and then generate $K' = h(S_1 \parallel S_2 \parallel \cdots \parallel S_n)$. If $proof_{PK_m} \neq h(K' \parallel g)$ abort, otherwise compute $U =_{1/K'} \{U_K\}$.

2.2 Threshold Cryptosystem

The concept of a threshold scheme, called secret sharing scheme was introduced in [17] and since then many researchers have investigated such schemes.

A (k, n) -threshold secret sharing scheme is a protocol among n players in which the *dealer* distributes partial information (*share*) about a *secret* to n participant such that: a) Any group of fewer k participants can not obtain any information about the secret. b) Any group of at least k participants can compute the secret in polynomial time.

We use the verifiable secret sharing (VSS) scheme by Pedersen [20], denoted as *Pedersen-VSS*. The next lemma summarizes some properties of *Pedersen-VSS*, which are used in the analysis of our protocol.

Lemma 1. ([20]) *Pedersen-VSS satisfies the following properties in the presence of an adversary that corrupts at most $k-1$ parties and which cannot compute $\log_g h$:*

1. *If the dealer is not disqualified during the protocol then all honest players hold shares that interpolate to a unique polynomial of degree $k-1$. In particular, any k of these shares suffice to efficiently reconstruct (via interpolation) the secret s .*
2. *The protocol produces information (the public values E_i and private value s_i) that can be used at reconstruction time to test for the correctness of each share; thus, reconstruction is possible, even in the presence of malicious players, from any subset of shares containing at least k correct shares.*
3. *The view of the adversary is independent of the value of the secret s , and therefore the secrecy of s is unconditional.*

In the next section, we describe our proposed password-based authentication scheme making use of the (k, n) -threshold scheme in which a user distributes secrets to multiple servers, assuming $n \geq 2k - 1$ [20,10].

2.3 Bilinear Pairings

Let us consider an additive group \mathbb{G}_1 and a multiplicative group \mathbb{G}_2 of the same order q . Assume that the discrete logarithm problem is hard in both groups. Let P be a generator of \mathbb{G}_1 , and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ a bilinear map satisfying the following properties:

1. *Bilinearity*: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$.
2. *Non-degeneracy*: if $\hat{e}(P, Q) = 1$ for all $Q \in \mathbb{G}_1$, then $P = \mathcal{O}$.
3. *Computability*: there exists an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

With such groups \mathbb{G}_1 and \mathbb{G}_2 , we can define hard cryptographic problems like *Computational Diffie-Hellman (CDH)* problem, *Decision Diffie-Hellman (DDH)* problem, and *Gap Diffie-Hellman (GDH)* problem. We skip detailed definition of these problems. To construct the bilinear pairing, we can use the Weil pairing and Tate pairing. \mathbb{G}_1 is a cyclic subgroup of the additive group of points of a supersingular elliptic curve $E(\mathbb{F}_p)$ over a finite field while \mathbb{G}_2 is a cyclic subgroup of the multiplicative group associated to a finite extension of \mathbb{F}_p .

3 Our Proposed Scheme

3.1 Model

Our model for multi-server roaming protocol is similar to that of [9], but with some different features. First, our scheme employs the concept of threshold scheme, where the user plays the role of a dealer to distribute secret shares to n servers. To do this, we make use of the Pedersen-VSS protocol [20] in a different way that only the user who knows an extra information, *password*, can obtain the secret value derived from the password in collaboration with threshold servers. While the protocol in [9] uses a n -out-of- n solution, *i.e.*, the password information is shared among n servers and they *all* must cooperate to authenticate the user, the protocol in our model uses k -out-of- n solution. In addition, even if an adversary compromises k or more servers, she cannot reconstruct the secret value, without knowing user's password. Second, our scheme is based on bilinear pairings that can be built from Weil pairing or Tate pairing over *GDH* group, which *CDH* problem is hard but *DDH* problem is easy. On the other hand, although we don't consider *forgiveness protocol* introduced in [9] by which user's honest mistakes are forgiven by sending evidence of recent prior invalid access attempts after each successful authentication, this forgiveness can be adapted in our system. Figure 1 depicts the concept of our model.

There are two phases: (1) Enrollment phase — The user, U enrolls his credentials in the servers at his own client terminal, C which may hold user's private information. (2) Retrieval phase — The user may move to other place where he is just able to use different client terminal, C' which does not hold any user-specific information. The protocol, however, allows the user to download a private key from remote servers with knowledge of only his identity and password.

Enrollment [TPS Protocol]. After constructing (k, n) -threshold system as similar as in Pedersen-VSS, the client terminal (For simplicity, we say "client".) creates n shares using the Pedersen-VSS scheme. k shares will contribute to reconstruct the master symmetric key K_m which is derived from a user's password π . Then,

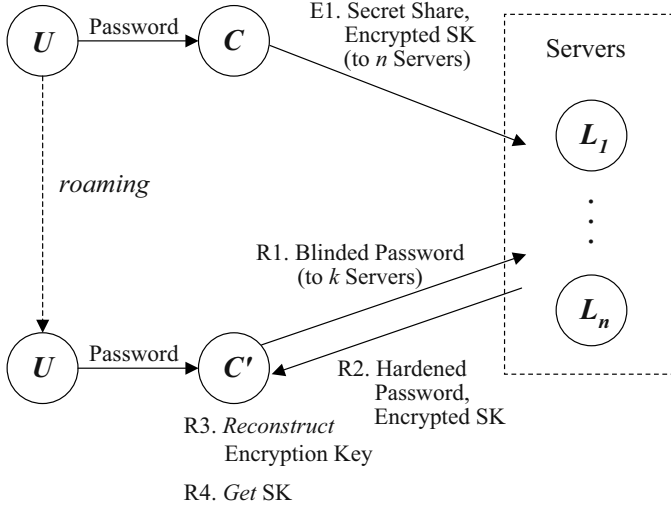


Fig. 1. The concept of our model

the client have user's private key SK encrypted with the symmetric key K_m . Finally, he creates a proof value V that links the password to his master key.

The client sends secretly share Y_i , the encrypted private key U_K , and the proof value V to each server.

As in [9], the enrollment protocol flow has to be performed through a secure channel that authenticates the identity of the user to each i^{th} server L_i .

Authenticated Retrieval [TPR Protocol]. When at any available client terminal, the user wants to download his private key stored in the server, the client first performs the threshold protocol with at least k servers. Note here that no client terminal has a user's information created at enrollment time.

The client randomly chooses at least k servers and sends them a randomly blinded form of the password to each server. On receiving the request, each server in turn responds with a blinded reply consisting of the blinded password. At least one of the servers also sends the encrypted private key U_K and its proof value V to the client.

The client reconstructs user's master key K_m using the shares and user's password, and then verifies whether the master key is correct using the proof value V and the master key K_m . Finally, if the master key is correct, the user gets his private key SK by decrypting U_K with the master key.

3.2 Definitions

Communication Model. We assume that our computation model is composed of a set of n players $\{L_1, L_2, \dots, L_n\}$. They are connected by a complete network of

secure point-to-point channels. The players have access to a dedicated broadcast channel in which when a player sends a broadcast message all other players can receive the message and know exactly from whom the message sent.

The Adversary. We assume that there exists an adversary, \mathcal{A} , who can corrupt up to $k - 1$ of n servers in the network, where $n \geq (2k - 1)$. We consider a malicious adversary that may cause corrupted players to divert from the specified protocol in any way. We assume that the computational power of the adversary is adequately modeled by a probabilistic polynomial time Turing machine. Our adversary is *static*, i.e., chooses the corrupted players at the beginning of the protocol.

Now, we state some definitions similar in [11,12], for the analysis of our protocol.

In the following we assume that there are a dealer C and n players $\{L_1, L_2, \dots, L_n\}$. We say that C **broadcasts** a message m , when she puts m on the broadcast channel for everybody to hear it. In particular \mathcal{A} can hear the message too. We say that C **distributes** a message if she puts m on the private channels connecting her to all the other players. Notice that \mathcal{A} can see m only if somebody has been corrupted.

Let \mathcal{P} be a pair of protocols where the second is always executed after the first one, $\mathcal{P} = (\text{Share-Verify}, \text{Recover})$ for the players $\{L_1, L_2, \dots, L_n\}$ and a dealer C .

Definition 1. (View) *The adversary view, $\text{View}_{\text{Network}, \mathcal{A}}^{\mathcal{P}}(\cdot)$ during protocol \mathcal{P} is the probability distribution over the set of computational histories (traffic and coin tosses) of the bad players.*

Sometimes we accompany some distributed protocol \mathcal{P} we propose by a description of a *simulator* Sim which is needed to analyze the security of this protocol. Sim is an algorithm that plays the role of the honest players. \mathcal{A} interacts with Sim as if she was interacting with the network. Sim tries to create a view for \mathcal{A} that is indistinguishable from the real one. That is, the process of simulation is a computation of two interactive algorithms, Sim and \mathcal{A} , where the simulator controls the uncorrupted players, and \mathcal{A} controls the corrupted players. Therefore a description of a simulation process is similar to a description of the protocol itself [13].

Definition 2. *The protocol \mathcal{P} is called k -secure (or secure with threshold k) if in the presence of an attacker that corrupts at most $k-1$ parties the following requirements for correctness and secrecy are satisfied:*

Correctness:

1. *All subsets of k shares provided by honest players define the same unique secret value.*
2. *Secret value is uniformly distributed.*

Secrecy: (Simulatability) *For every (probabilistic polynomial-time) adversary \mathcal{A} , there exists a (probabilistic polynomial-time) simulator Sim , such that on input an element Y , produces an output distribution which is polynomially indistinguishable from \mathcal{A} 's view of a run of \mathcal{P} that ends with Y as its output, and where \mathcal{A} corrupts up to $k-1$ parties. That is, the adversary view $\text{View}_{\text{Network}, \mathcal{A}}^{\mathcal{P}}(\cdot)$ is identical with $\text{View}_{\text{Sim}, \mathcal{A}}^{\mathcal{P}}(\cdot)$ which is the adversary view of the simulated execution of the protocol.*

A simulator of each subprotocol exhibits the secrecy of this subprotocol, which states that the adversary learns nothing from the protocol beyond the public inputs and outputs of this protocol, or in other words, that the adversary learns as much by participating in the threshold computation as he would learn from observing this operation as a block-box.

We now come up with the following definition of the secure threshold password-authenticated key retrieval protocol (TPKR for short).

Definition 3. *In our $\text{TPKR} = (\text{TPS}, \text{TPR})$, Let two kinds of (static) adversaries exist as follows:*

1. *Strong Adversary: the adversary is able to corrupt and to control k or more servers if he desires.*
2. *Weak Adversary: Not strong adversary, i.e., the adversary is just able to corrupt at most $k-1$ servers.*

Definition 4. (Secure TPKR) *Let TPKR be the (k, n) -threshold password-authenticated key retrieval protocol, where $2k-1 \leq n$. TPKR is a strongly secure protocol if:*

1. *The protocol is k -secure satisfying Definition 2 in the presence of the weak adversary in Definition 3.*
2. *No adversary, even strong one, without knowing user's password is able to reconstruct the master symmetric key K_m , and is thus able to obtain the private key SK .*

3.3 Detailed Protocol

We let ℓ be the security parameter given to the setup algorithm and let \mathcal{G} be some GDH parameter generator.

System Setup Given a security parameter ℓ , the algorithm \mathcal{G} works as follows:

1. Run \mathcal{G} on input ℓ to generate a prime $q \geq 2^\ell$, two cyclic groups \mathbb{G}_1 and \mathbb{G}_2 of the same order q and a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
2. Choose two arbitrary generators P and $P' \in \mathbb{G}_1$, where $P' = \alpha P$ for some $\alpha \in \mathbb{Z}_q$ and the computing α given P and P' is infeasible.
3. Choose cryptographic hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^\kappa$, and $H_3 : \{0, 1\}^\kappa \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$, for some κ . The security analysis will view H_1, H_2 , and H_3 as random oracles.
4. The system parameters are $\text{params} = \{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, H_1, H_2, H_3, P, P'\}$. And then publish them.

Enrollment Protocol \mathcal{TPS} The enrollment protocol makes use of Pedersen-VSS protocol, but we introduce elliptic curve cryptosystems for bilinear parings.

Denote n servers involving in the protocol as $\{L_1, L_2, \dots, L_n\}$ and the client playing the role of a dealer as C . Let ID and SK be the user's identity and the private key, respectively. The user having ID picks his password π . We assume that π can be mapped into \mathbb{Z}_q . The user then performs the following protocol at the client terminal as follows:

1. The client C , as a dealer, distributes user's credentials.
 - (a) Select randomly y and $z \in \mathbb{Z}_q^*$.
 - (b) Choose two random polynomials $f(x)$ and $g(x)$ over \mathbb{Z}_q of degree $k-1$ such that $f(0) = a_0 = y$ and $g(0) = b_0 = z$. Let

$$\begin{aligned} f(x) &= a_0 + a_1x + \dots + a_{k-1}x^{k-1} \quad \text{and} \\ g(x) &= b_0 + b_1x + \dots + b_{k-1}x^{k-1}. \end{aligned}$$

- (c) Compute and broadcast $E_i = E(a_i, b_i) = a_iP + b_iP'$ for $i = 0, \dots, k-1$.
 - (d) Compute $K = \hat{e}(yR_{ID}, Q_{ID})$, then create the master symmetric key $K_m = H_2(K)$, where $R_{ID} = H_1(\pi)$ and $Q_{ID} = H_1(ID)$. Then create the encrypted private key $U_K = E_{K_m}(SK)$ and the key verifier $V = H_3(K_m, P)$, and let $Y_i = f(i) \cdot Q_{ID}$ and $Z_i = g(i) \cdot Q_{ID}$ for $i = 1, 2, \dots, n$.
 - (e) Send $\{ID, Y_i, Z_i, U_K, V\}$ *secretly* to each player L_i for all $i \in [1, n]$.
2. After receiving all the information from C , L_i does as follows.
 - (a) Verifies that

$$\hat{e}(Y_i, P) \cdot \hat{e}(Z_i, P') \stackrel{?}{=} \hat{e}\left(Q_{ID}, \sum_{j=0}^{k-1} i^j \cdot E_j\right). \quad (1)$$

- (b) If Eq.(1) is verified to be false, response a *complaint* against C . Otherwise accept and store $\{ID, Y_i, U_K, V\}$ in a storage maintained on each L_i .
3. C discards all information, and completes the enrollment protocol.

For the sake of convenience, we assume that the client has received no complaint in Step 2.

Authenticated Retrieval Protocol \mathcal{TPR} For authenticated retrieval, the client and k servers perform the actions as follows: Denote k servers by $B = \{L_i \mid 1 \leq i \leq k\}$.

1. C sends k servers a request message.
 - (a) Select a random number uniformly distributed $x \in \mathbb{Z}_q^*$, compute $X = xR_{ID}$.
 - (b) Send X and ID to each server L_i for $i \in B$.

2. On receiving the request, each server L_i responds as follows:
 - (a) Retrieve $\{ID, Y_i, U_K, V\}$ from the storage maintained securely on each L_i .
 - (b) Compute $R_i = \hat{e}(X, Y_i)$, and then reply $\{R_i, U_K, V\}$ to the client.
3. Finally, the client reconstructs user's private key by performing the following:
 - (a) Compute $l_i = \prod_{j \in B, j \neq i} \frac{j}{j-i}$ for each i^{th} server.
 - (b) Compute $R'_i = (R_i)^{l_i x^{-1}}$ and $K' = \prod_{i \in B} R'_i$.
 - (c) Generate $K'_m = H_2(K')$.
 - (d) If $V \neq H_3(K'_m, P)$, abort.
 - (e) To obtain the private key, decrypt U_K with the master key K'_m .

Completing the protocol successfully, the client reconstructs the user's private key SK . Figure 2 depicts the retrieval protocol.

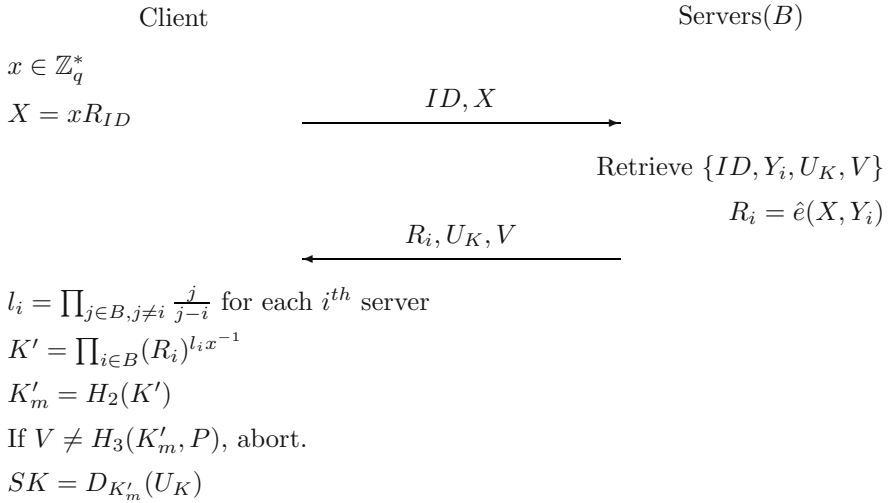


Fig. 2. Our Threshold Key Retrieval Protocol

4 Security Analysis

In this section, we discuss with the security aspects of our proposed scheme TPKR and give the complete security proof. As a result, the security for our protocol arrives at the security for secret value K , assuming that the adapted symmetric cryptosystem is secure and thus nobody can obtain the private key SK without knowing K . Note that $UK = E_{K_m}(SK)$, where $K_m = H_2(K)$.

Lemma 2. (Correctness) *The protocol $TPKR = (TPS, TPR)$ from Section 3.3 satisfies the correctness of Definition 2 with threshold k , for any $k \leq (n+1)/2$.*

Proof. First note that all the honest players indeed hold the verification equation Eq.(1) as follows:

$$\hat{e}(Y_i, P) \cdot \hat{e}(Z_i, P') = \hat{e}\left(Q_{ID}, \sum_{j=0}^{k-1} i^j \cdot E_j\right).$$

Since

$$\hat{e}(Y_i, P) \cdot \hat{e}(Z_i, P') = \hat{e}(f(i)Q_{ID}, P) \cdot \hat{e}(g(i)Q_{ID}, P') = \hat{e}(Q_{ID}, f(i)P + g(i)P'),$$

where

$$f(i)P + g(i)P' = \sum_{j=0}^{k-1} i^j (a_j P + b_j P') = \sum_{j=0}^{k-1} i^j E_j.$$

1. From **Lemma 1.1**, we know that all honest players hold shares (Y_i) which contribute to reconstruct unique secret by combining with client's request message as in step 2 of \mathcal{TPR} protocol. For any set \mathcal{B} of k shares and extra value (X) from client's request message, the unique secret is computed as follows:

$$\begin{aligned} K' &= \prod_{i \in \mathcal{B}} \hat{e}(X, Y_i)^{l_i x^{-1}} = \prod_{i \in \mathcal{B}} \hat{e}(xR_{ID}, f(i)Q_{ID})^{l_i x^{-1}} = \prod_{i \in \mathcal{B}} \hat{e}(f(i)l_i R_{ID}, Q_{ID}) \\ &= \prod_{i \in \mathcal{B}} \hat{e}(f(i) \prod_{j \in \mathcal{B}, j \neq i} \frac{j}{j-i} R_{ID}, Q_{ID}) = \hat{e}(\sum_{i \in \mathcal{B}} f(i) \prod_{j \in \mathcal{B}, j \neq i} \frac{j}{j-i} R_{ID}, Q_{ID}) \\ &= \hat{e}(yR_{ID}, Q_{ID}), \end{aligned}$$

where l_i is an appropriate Lagrange interpolation coefficient for the set \mathcal{B} . Since the above holds for any set of k correct shares then K' is uniquely defined, where the same extra value (X) which as said is derived from the user's password has to be given to the protocol \mathcal{TPS} and \mathcal{TPR} .

2. Let's consider the secret value K . We can let K be $g^{\lambda y}$ for some λ where g is a generator of group \mathbb{G}_2 . Since y is chosen randomly from \mathbb{Z}_q^* as in [20], therefore $K = g^{\lambda y}$ is also a random element in the group \mathbb{G}_2 . On the other hand, from **Lemma 1.3**, the view and thus actions of the adversary are independent of the secret y .

As a result, we can state that TPKR can be resistant to corruption of even $k-1$ of $n \geq 2k-1$ servers. A user chooses randomly a secret value y uniformly distributed in \mathbb{Z}_q^* during the execution of \mathcal{TPS} . Even there exists an adversary who can corrupt at most $k-1$ servers among $n \geq 2k-1$, any subset of k servers constructs the secret value K uniformly distributed in \mathbb{G}_2 .

Lemma 3. (Secrecy) *Protocol TPKR from Section 3.3 satisfies the secrecy of Definition 2.*

Proof. We can prove in a similar way which is used to prove Lemma 3 in [13]. We can state that, for any input secret y and y' , if the dealer \mathcal{C} is uncorrupted then there is no difference between the adversarial view of an execution of TPKR in which \mathcal{C} shares y' , from a view of TPKR in which \mathcal{C} shares y .

There exists Sim such that for every $n/2$ -threshold static secure-channels adversary \mathcal{A} with history $h_{\mathcal{A}}$, for any given system parameter **params**. Let E stand for an instance (P, P') of Pedersen commitment [20]. Let $f(x), g(x)$ be any $k-1$ degree polynomials such that $f(0) = y$. Consider a run of TPKR in which \mathcal{C} uses polynomials $f(x), g(x)$ and the random input of \mathcal{A} is $r_{\mathcal{A}}$. Note that once we fix $f(x), g(x), r_{\mathcal{A}}$ then everything else in the run of this protocol is determined. Denote the outputs of such run as $\mathcal{TPKR}_{\mathcal{A}}^{Data}((h_{\mathcal{A}}, r_{\mathcal{A}}), E; f(x), g(x))$. We denote the set of corrupted players as $P_{\mathcal{B}}$ and the set of uncorrupted players as $P_{\mathcal{G}}$.

Note that any $k-1$ degree polynomials $f'(x), g'(x)$ such that $f'(i) = f(i)$ for $L_i \in P_{\mathcal{B}}$ and $f(x) + \alpha g(x) = f'(x) + \alpha g'(x)$ where $P' = \alpha P$, the adversary's output in $\mathcal{TPKR}_{\mathcal{A}}^{Data}((h_{\mathcal{A}}, r_{\mathcal{A}}), E; f'(x), g'(x))$ is the same as in $\mathcal{TPKR}_{\mathcal{A}}^{Data}((h_{\mathcal{A}}, r_{\mathcal{A}}), E; f(x), g(x))$.

If we fix $y, y', r_{\mathcal{A}}$, and range the polynomials $f(x), g(x)$ among all $k-1$ degree polynomials such that $f(0) = y$, then we see that the distribution of the adversary view in the following two cases are equal, for every y, y' , and $r_{\mathcal{A}}$:

1. TPKR on $f'(x), g'(x)$, and $r_{\mathcal{A}}$ followed by protocol on the resulting $\mathcal{TPKR}_{\mathcal{A}}^{Data}$, where $f'(x), g'(x)$ are random $k-1$ degree polynomials such that (0) $f'(0) = y'$; (1) $f'(i) = f(i)$ on $L_i \in P_{\mathcal{B}}$; (2) $f'(x) + \alpha g'(x) = f(x) + \alpha g(x)$ where $f(x), g(x)$ are random $k-1$ degree polynomials such that $f(0) = y$.
2. TPKR on $f(x), g(x)$, and $r_{\mathcal{A}}$ with outputs denoted $\mathcal{TPKR}_{\mathcal{A}}^{Data}[y]$, followed by protocol on inputs $\mathcal{TPKR}_{\mathcal{A}}^{Data}[y]$ output by replacement procedure $\mathcal{T}_{TPKR}(\mathcal{TPKR}_{\mathcal{A}}^{Data}[y], y', \alpha)$ where $f(x), g(x)$ are random $k-1$ degree polynomials such that $f(0) = y$. $\mathcal{T}_{TPKR}(\mathcal{TPKR}_{\mathcal{A}}^{Data}[y], y', \alpha)$ mentioned above takes as inputs a given secret-sharing $\mathcal{TPKR}_{\mathcal{A}}^{Data}[y]$, target value y' and α s.t. $P' = \alpha P$, and outputs its replacement $\mathcal{TPKR}_{\mathcal{A}}^{Data}[y']$ as in [13]. Note that the data which is visible to \mathcal{A} , i.e., the public data and the private data of the players controlled by the \mathcal{A} , must remain the same in $\mathcal{TPKR}_{\mathcal{A}}^{Data}[y]$ and $\mathcal{TPKR}_{\mathcal{A}}^{Data}[y']$, so this *replacement* is always only a modification of the private data of the players controlled by the simulator.

From the above observations, (1) since $f(x)$ is a random polynomial such that $f(0) = y$, then $f'(x)$ is a random polynomial such that $f'(0) = y'$; and (2) there is a one-to-one mapping between a choice of $g(x)$ and a choice of $g'(x)$, and thus since $g(x)$ is a random polynomial then so $g'(x)$.

From Lemmas 2 and 3, we can state the following theorem:

Theorem 1. Assume that $n \geq 2k - 1$. Then the protocol TPKR is k -secure threshold-authenticated roaming protocol according to Definition 2 with fault-tolerance k .

The following lemma shows the security of the protocol \mathcal{TPR} against a strong adversary who corrupts k or more servers if he desires.

Lemma 4. *Under GDH assumption, the protocol \mathcal{TPR} is secure against a strong adversary defined by Definition 3.*

Proof. Let a *strong adversary* \mathcal{A} be able to corrupt k or more servers and thus to obtain at least k shares. In this case, we need to show that \mathcal{A} can not reconstruct the secret value K' without knowing the user's password.

In order to break the protocol, \mathcal{A} tries to compute K'' such that

$$K'' = \prod_{i \in S} \hat{e}(R', Y_i)^{l_i}, \quad (2)$$

with knowledge of system parameter **params**, any set \mathcal{S} of t secret shares Y_i for $i = 1, 2, \dots, t$ ($t \geq k$) and client's request messages X , where l_i is an appropriate Lagrange interpolation coefficient for the set \mathcal{S} .

We assume \mathcal{A} has three options to compute K'' : (1) Solve DLP, *i.e.* find an integer n such that $Q = nP$, (2) Solve CDHP in such a way that given $(P, \alpha P, \beta P) \in \mathbb{G}_1$ compute $\alpha\beta P$, and (3) Guess correct password, *e.g.*, by mounting password guessing attack.

1. In order to compute Eq.(2), \mathcal{A} first unblinds X to obtain R' , *i.e.* find an integer x (or $x' = x\beta$) such that $R' = xR_{ID}$ (or $R' = x'P$). Thus no adversary can compute R' , under the assumption DLP is hard.

2. Let $Q_{ID} = \alpha P$, $R_{ID} = \beta P$. Even given a triple $(P, \alpha P, y\beta P)$, \mathcal{A} can not compute $\alpha\beta yP$, such that $K'' = \hat{e}(yR_{ID}, Q_{ID}) = \hat{e}(P, P)^{\alpha\beta y} = g^{\alpha\beta y}$ where g is a generator of \mathbb{G}_2 , assuming that \mathbb{G}_1 is a GDH group.

On the other hand, given $\{Y_i = y_i Q_{ID} \mid i = 1, 2, \dots, t\}$, \mathcal{A} can compute the following at the best:

$$Y'' = \prod_{i \in S} \hat{e}(y_i Q_{ID}, P)^{l_i} = \hat{e}(y Q_{ID}, P) = \hat{e}(P, P)^{\alpha y} = g^{\alpha y}.$$

3. Let π' be a password guess by \mathcal{A} . Thus $R' = H_1(\pi')$. Now, \mathcal{A} computes the following:

$$K'' = \prod_{i \in S} \hat{e}(R', Y_i)^{l_i} = \prod_{i \in S} \hat{e}(R', f(i)Q_{ID})^{l_i} = \hat{e}(yR', Q_{ID}),$$

However, \mathcal{A} can not verify whether his guess is correct or not. More over, by Lemma 3 it is impossible to distinguish K'' from K' .

Theorem 2. *Assume $n \geq 2k - 1$. Then the protocol \mathcal{TPKR} is strongly secure according to Definition 4, under GDH assumption.*

Proof. The proof of the theorem comes immediately from Theorem 1 and Lemma 4.

5 Comparison

With mainly compared to [6] and [9], our scheme is capable of resisting that fewer than threshold servers are compromised. When only k honest servers are involved in the protocol, the user can retrieve his private key with knowledge of his own password. Besides, even attacker has succeeded in compromising k or more servers but without knowing the user's password, she still cannot obtain any information about the user's credential.

Table 1 depicts computation load of TPKR compared with [9]. We denote \mathbf{E} and \mathbf{M} by computation load for exponentiation and multiplication, respectively. Let n be the number of servers and k be threshold value.

Table 1. Computation in the Retrieval on the Client Side

	Jab01 [9]	TPKR
Main computation parts	$S_i = R_i^{x^{-1}}$ $K' = h(S_1 \parallel \cdots \parallel S_n)$	$R'_i = R_i^{l_i x^{-1}}$ $K' = h(\prod_{i \in B} R'_i)$
Computation load	$n\mathbf{E}$	$k(\mathbf{E}+\mathbf{M})$

From Table 1, we see that our scheme TPKR is more efficient one than *Jab01* with respect to the computation during the retrieval, since the inequality $n\mathbf{E} \geq k\mathbf{E}+k\mathbf{M}$ holds, where $n \geq 2k - 1$, *i.e.*, $(k - 1)\mathbf{E} \geq k\mathbf{M}$ if $k \geq 2$. However with respect to the server side, the computation load of our protocol may be less efficient due to pairing operation which costs several times expensive than the exponentiation [21].

6 Conclusions

Based on pairings, we firstly suggest a new threshold password-only roaming protocol which allows a roaming user to download a private key from remote servers, without revealing the password to off-line guessing. No client terminal has his information created at enrollment time.

We note that, as a goal of a multi-server roaming system, a protocol has to allow a user to get his private key from the servers, even if some of the servers are compromised. In this paper, we use (k,n) -threshold scheme in which only k honest servers or more are engaged to reconstruct a secret key based on bilinear pairings that could be built from Weil pairing or Tate pairing. The protocol is useful for authenticating roaming users and even non-roaming users, and retrieving private keys for use in other applications. A design of more efficient schemes based on parings is left as one of further works.

Acknowledgement

The authors are very grateful to the anonymous reviewers for their valuable suggestions and comments to this paper.

References

1. S.Al-Riyami and K.Paterson, "Certificateless Public Key Cryptography", available at <http://www.ime.usp.br/~rt/cranalysis/CertifLessPKC.pdf>, Jul.2003.
2. D.Boneh and M.Franklin, "Identity-Based Encryption from the Weil Pairing", *CRYPTO2001*, LNCS 2139, pp.213-229, Springer-Verlag, 2001.
3. S.Bellovin and M.Merritt, "Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks", *Proc. of IEEE Symposium on Research in Security and Privacy*, May 1992.
4. S.Bellovin and M.Merritt, "Augmented Encrypted Key Exchange: A Password-based Protocol Secure Against Dictionary Attacks and Password File Compromise", Technical Report, AT&T Bell Laboratories, 1994.
5. J.Baek and Y.Zheng, "Identity-Based Threshold Decryption", *IACR eprint*, 2003/164.
6. W.Ford and B.Kaliski, "Server-Assisted Generation of a Strong Secret from a Password", *Proc.9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, IEEE, Jun.14-16, 2000.
7. F.Hess, "Efficient Identity Based Signature Schemes Based on Pairings", *SAC2002*, LNCS 2595, pp.310-324, Springer-Verlag, 2003.
8. D.Jablon, "Strong Password-Only Authenticated Key Exchange", *ACM Computer Communications Review*, Oct.1996.
9. D.Jablon, "Password Authentication Using Multiple Servers", *CT-RSA2001*, LNCS 2020, pp.344-360, Springer-Verlag, 2001.
10. B.Libert and J.Quisquater, "Efficient Revocation and Threshold Pairing Based Cryptosystems", *PODC'03*, pp.163-171, Jul.13-16, 2003.
11. R. Gennaro, "Theory and Practice of Verifiable Secret Sharing", PhD Thesis, MIT, May 1996.
12. R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems", *Advances in Cryptology - EURO-CRYPT '99*, LNCS 1592, pp.295-310, Springer-Verlag, 1999.
13. S. Jarecki, "Efficient Threshold Cryptosystems", PhD Thesis, MIT, June 2001.
14. P.MacKenzie, T.Shirmpston and M.Jakobsson, "Threshold Password-Authenticated Key Exchange(Extended Abstract)", *CRYPTO2002*, LNCS 2442, pp.385-400, Springer-Verlag, 2002.
15. T.Pedersen, "Non-interactive and Information Theoretic Secure Verifiable Secret Sharing", *CRYPTO'91*, LNCS 576, pp.129-140, Springer-Verlag, 1992.
16. R.Perman and C.Kaufman, "Secure Password-Based Protocol for Downing a Private Key", *Proc. 1999 Network and Distributed System Security Symposium*, Internet Society, Jan.1999.
17. A.Shamir, "How to Share a Secret", *Communication of the ACM*, Vol.22, No.11, pp.612-613, Nov.1979.
18. D.Vo, F.Zhang and K.Kim "A New Threshold Blind Signature Scheme from Pairings", *SCIS2003*, Vol.1/2, pp.233-238, Jan.2003.
19. T.Wu, "The Secure Remote Password Protocol", *Proc. of Network and Distributed System Security Symposium*, pp.97-111, Internet Society, Jan.1998.
20. T. Pedersen, "Non-interactive and Information theoretic Secure Verifiable Secret Sharing", *Advances in Cryptology - CRYPTO '91*, LNCS 576, pp.129-140, Springer-Verlag, 1992.
21. J. Cha and J. Cheon, "An Identity-Based Signature from Gap Diffie-Hellman Groups", *PKC 2003*, LNCS 2567, pp.18-30, Springer-Verlag, 2003.

An Efficient Group Key Management Scheme for Secure Multicast with Multimedia Applications

Chang N. Zhang¹ and Zheng Li²

¹ Department of Computer Science, University of Regina,
Regina, SK, S4S 0A2, Canada
zhang@cs.uregina.ca

² Department of Computer Science, University of Toronto,
Toronto, ON, M5S 3G4, Canada
zli@cs.utoronto.ca

Abstract. In this paper we present an efficient and scalable group key management scheme for secure one-to-many multicast. The proposed scheme has several advantages over existing protocols. It supports dynamic subgrouping and does not require reliable multicast service to provide key distribution. In addition, a one-way hash function is adopted to handle massive member join and move requests efficiently without secret key transmissions and encryption/decryption computations. We also demonstrate how to deploy fast MPEG encryption algorithms and digital rights management systems to the scheme. The proposed group key management scheme is especially suitable for delay-sensitive multimedia multicast applications and resource-constraint equipments

Keywords: Secure Multicast, Key Management, Copyright Protection, Digital Rights Management, Multimedia Data Security.

1 Introduction

Multicast [4] is an inter-network service which provides efficient delivery of data from a single source to a group of recipients. It reduces the bandwidth requirements of the network and the computational overhead of the host devices. This makes multicast an ideal technology for communication amongst a large group of participants [7].

Secure multicast communications include many types of service, such as pay-per-view, video conferencing, real-time delivery of stock quotes, etc. The objectives of secure multicast communications are to preserve authentication and secrecy for all group communications so that group participants may access the distributed information only when authorized and to ensure that only those authorized can distribute information to the group. These security requirements are fulfilled by the cryptosystem, which requires a group key management solution to distribute and maintain cryptographic keys within registered group members, who can then use these keys to encrypt or decrypt the multicast data. Because

multicast group membership is dynamic, when a new member joins or an existing member leaves the multicast group, group keys must be updated accordingly to achieve *membership forward secrecy* and *membership backward secrecy* [12].

Existing secure multicast protocols (group key management schemes) can be categorized into the following: *centralized group control approach*, in which only one central group controller exists; *subgroup control approach*, where the management of the group is divided amongst subgroup controllers; and *member control approach*, in which no explicit group controllers exist and the key generation and distribution are done by the members themselves. Each of the above classes is further divided into *scalable* and *non-scalable* schemes [7]. In general, the scalable subgroup control approach performs better than the scalable member control approach. Furthermore, the performance gain of the scalable subgroup control approaches increases with the multicast group size [5].

In this paper we investigate the issues of designing an efficient and scalable group key management scheme for dynamic groups. The idea is using a one-way hash function to handle massive member join operations, which eliminates secret key transmissions and encryption/decryption computations. Then, dynamic subgrouping and key redistribution methods are proposed based on the efficient member join method. The proposed scheme is falling into the scalable subgroup control approach and applicable to secure one-to-many multicast.

The paper is organized as follows. Section 2 outlines basic techniques for the scalable subgroup control approach. Section 3 presents an efficient and scalable group key management scheme. Section 4 provides detailed security and benefits analysis of the proposed scheme. Section 5 applies the proposed scheme to secure multimedia multicast. Section 6 concludes this paper.

2 The Scalable Subgroup Control Approach

There exist several secure multicast protocols in the scalable subgroup control approach, such as Iolus [14], the Nortel framework [8, 9], and the Dual Encryption Protocol (DEP) [6]. Below we briefly review some general techniques used by these protocols.

2.1 Hierarchical Subgrouping

The main idea for the scalable subgroup control approach is to partition a multicast group into a hierarchy of subgroups and for each subgroup to be managed by a subgroup controller (SC). Each SC is responsible for two tasks: managing its subgroup keys and mediating all communication between its subgroup and other subgroups. SCs and subgroups can exist on multiple levels, and lower-level SCs are clients of higher-level SCs. Figure 1 presents an example of a hierarchical secure one-to-many multicast tree. For simplicity, we adopt a 3-level secure multicast tree in this paper.

2.2 Member Join and Leave

The procedure for a new member to join subgroup i is as follows: SC_i generates a new subgroup key; then, SC_i negotiates a unique *unicast key* with the new member; SC_i sends the new subgroup key to the new member using the unicast key (via the secure unicast channel [10]); for other subgroup members, SC_i multicasts the new subgroup key to its subgroup by encrypting it under the old subgroup key; current members decrypt the multicast message with the old subgroup key and thus obtain the new key.

If a member is leaving subgroup j , the procedure is: SC_j generates a new subgroup key and creates a large message containing n copies of the new subgroup key (assuming n remaining members in subgroup j); each copy is encrypted with the unicast key shared between the SC and a remaining member; then SC_j multicasts this message to its subgroup; those members who stay in subgroup j can decrypt specific portions of the multicast message using their unicast keys.

2.3 Data Transmission and Authentication

Since each subgroup uses a different subgroup key, the SC is responsible for translating encrypted data from its parent subgroup key to its child subgroup key and multicasting it to its child subgroup. There are two types of SCs: fully-trusted SCs and partially-trusted SCs. Fully-trusted SCs have the access to the transmitted multicast payload data. Examples are Iolus and the Nortel framework. However, in some applications, SCs are only responsible for managing its subgroup membership and not eligible to access payload data. For such partially-trusted SCs, a dual encryption can hide payload data from them, e.g. in DEP, where multicast payload data have been encrypted twice in a row with two different keys, and one of the keys is unobtainable to SCs.

The power of multicast could let an attacker send malicious packets to millions of receivers. Therefore, receivers need multicast source authentication to verify that a given packet originates from the correct source. The conventional approach to achieve authentication is to use public-key cryptography, e.g. digital signatures. Unfortunately, digital signatures have a high computation overhead for both the sender and the members, and they also involve a high communication overhead [15].

2.4 Issues

The following issues need to be resolved in the subgroup control approach:

- a) Both the member join and leave methods use multicast to distribute keying material to multicast members. However, in some applications, the multicast service such as IP multicast is unreliable [4]. Unreliable multicast may cause group members to miss the latest keying material, which would then cause transient security breaches. Therefore, efficient key redistribution is critical to a group key management scheme.

- b) In some multicast applications, the multicast group has highly dynamic membership changes, i.e. members join and leave multicast group frequently. This will result in flurries of key update messages that increase the probability of transient security breaches and confusion [14]. As we just mentioned above, if the underlying multicast service is unreliable, high membership dynamics may cause many members to miss the latest keying material, which then will trigger massive key redistribution operations. In addition, at peak time (e.g. before a pay-per-view movie multicast session) a high volume of sign-on (member join) requests can be expected. Thus, an efficient scheme to handle massive member join requests is necessary.
- c) In the subgroup control approach, the scalability heavily depends on how well the members are actually distributed among the subgroups. One of the solutions is to use “dynamic subgrouping”; that is, the membership of the overloaded SC is split and reallocated between itself and other SCs. A simple method to handle dynamic subgrouping is applying a member leave operation followed by a member join operation. Since secure multicast payload data cannot be communicated during the re-keying process, an important requirement for a key management scheme is to minimize the interruption in secure payload communications [3]. The simple method is therefore not suitable for frequent dynamic subgrouping operations.

3 The Proposed Group Key Management Scheme

The proposed group key management scheme involves the following four sets of entities: (1) the sender, responsible for encrypting and distributing multicast payload data to the multicast group, and managing SCs; (2) group members; (3) key distributors (KDs), in charge of distributing partial keying material to group members; (4) subgroup controllers (SCs), supervised by the sender to handle the workload of managing subgroups (i.e. re-keying subgroup keys when membership changes, and forwarding multicast payload data to their subgroups). KDs and SCs are third-parties, not members of the multicast group. As shown in Figure 1, subgrouping of multicast members is adopted in this paper to address scalability.

We choose the dual encryption method for data transmission, i.e. two keys are used to protect multicast payload data. The first key is called the group key (GK). GK is created by the sender and distributed to group members by KDs. The second key is a subgroup key (SK_i^m). Because each subgroup shares a different subgroup key, SK_i^m is only known to SC_i and its subgroup, where m is the value of the key counter. Each new key SK_i^m is generated by applying a one-way hash function F to the previous key SK_i^{m-1} . This means that $SK_i^m = F^m(SK_i^0)$, where SK_i^0 is the current key seed in subgroup i and $F^m(x) = F(F^{m-1}(x))$. Each key seed has a unique key identifier (key-ID). Table 1 overviews some notations used in this section ³.

³ In order to minimize the storage overhead, the one-way hash function F is used for three applications in the proposed group key management scheme: to generate new subgroup keys when new members join the multicast group; to calculate the Message

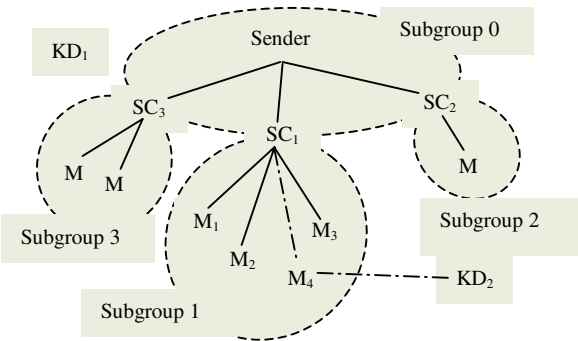


Fig. 1. A three-level secure one-to-many multicast tree with a new member join

Cryptographic Operations	$D_k(C)$: to decrypt ciphertext C using the secret key k
	$E_k(M)$: to encrypt plaintext M using the secret key k
	$F(M)$: to apply one-way hash function F on message M
Entities	KD_i : Key Distributor i
	M_i : Member i
	SC_i : Subgroup Controller i
Keys	DEK : Data Encryption Key
	GK : Group Key
	SK_i^m : subgroup key in subgroup i
	SK_i^0 : key seed of SK_i^m
	Key-ID: Key Identifier of SK_i^0

Table 1. Notations used in this paper

At system startup time, the sender makes all KDs be aware of its public key and the group key GK via secure unicast. Each SC generates a random number as the key seed (with a unique key-ID) to be used in its subgroup. SCs also clear their key counters to zero.

3.1 Member Join

In order to join the multicast group, a new member needs to obtain digital credentials from the sender. Each credential is signed by the sender to assert that the membership is valid for a given period of time. Sample credentials can be found in [6].

Authentication Code (MAC) value of each multicast packet by using Hashed MAC [11]; and to verify the disclosed MAC key belonging to the one-way key chain in the Timed Efficient Stream Loss-tolerant Authentication (TESLA) protocol [15, 16].

If there is an SC that has enough resources to accept a new member, the SC will accept her to its subgroup; otherwise, a new SC will be assigned ⁴ by the sender. For example, new member M_4 is going to join the secure multicast group (see Figure 1):

- (1) M_4 first contacts the nearest available KD (e.g. KD_1) and presents the credential. If M_4 's membership is valid, KD_1 sends GK to M_4 by secure unicast. M_4 then contacts the nearest SC (e.g. SC_1) to join its subgroup. After verifying M_4 's credential, SC_1 needs to revoke⁵ the credential instantly to prevent duplicate joins by M_4 . Therefore, the digital credentials adopted in the proposed scheme are one-time-use.
- (2) SC_1 suspends multicast payload data transmission; then, SC_1 applies one-way hash function F on current subgroup key SK_1^m and obtains SK_1^{m+1} : $SK_1^{m+1} = F(SK_1^m)$. SC_1 also increases key counter from m to $m+1$. Finally, SC_1 records the start time to use this new subgroup key, namely TK_1 .
- (3) SC_1 negotiates a unique unicast key with M_4 and then adds M_4 to its Current Members List (CML). CML is composed of member identifier, member's public key, the unicast key, membership start time and end time for each member who currently resides in this subgroup. SC_1 sends new subgroup key SK_1^{m+1} , the key counter, and the unique key-ID of the key seed SK_1^0 to M_4 via secure unicast (using the shared unicast key).
- (4) SC_1 creates and multicasts a Member Join Notification (MJN) message to its subgroup. The MJN message contains a member join flag, the key-ID of SK_1^0 , and the current key counter (see Table 2). Since we do not need confidentiality on the MJN message, it is sent by SC_1 in plaintext. Three existing members, M_1 , M_2 and M_3 , compare their local key-IDs with the key-ID in the received MJN message. If key-IDs are the same, three members apply function F once on their stored subgroup key SK_1^m and obtain SK_1^{m+1} : $SK_1^{m+1} = F^{(m+1)-m}(SK_1^m)$. They also increase their key counters from m to $m+1$. If the received key-IDs are not the same with member(s)' stored key-IDs, they may ask SC_1 to re-send the latest keying material.

Member join flag	Key-ID	Key counter
------------------	--------	-------------

Table 2. The Member Join Notification (MJN) message

- (5) SC_1 continues multicast payload data transmission, encrypted with the new subgroup key SK_1^{m+1} .

⁴ The member join, member leave, and key redistribution methods which will be described in this section are applicable to both group members and SCs. In other words, the sender manages subgroup 0 in the same way that SC_i manages its subgroup i .

⁵ We use a revocation list, which is used in Public-Key Infrastructure (PKI), to record the credentials that have been revoked. The revocation list is shared by all SCs.

3.2 Member Leave

For example, M_1 in subgroup 1 is leaving the multicast group and members M_2 , M_3 , and M_4 stay in subgroup 1. The procedure is as follows:

- (1) M_1 notifies SC_1 that she will leave the multicast group. Or, SC_1 expels M_1 from its subgroup when M_1 's membership expires. SC_1 suspends the multicast payload data transmission.
- (2) SC_1 sends a multicast message to subgroup 0 to let all other SCs know that M_1 is leaving the multicast group. The objective of this procedure will be discussed in the Member Move operation.
- (3) SC_1 generates a new random number as the new subgroup key seed SK_1^0 (with a unique key-ID) to replace the current one. SC_1 also clears its key counter to zero and deletes M_1 from its CML.
- (4) SC_1 sends a Member Leave Notification (MLN) message to remaining members. The message contains 3 copies of the member leave flag, new SK_1^0 , the key-ID of the new SK_1^0 , and the key counter (value is 0). Each copy is encrypted with the unicast key shared between SC_1 and a remaining member (i.e. M_2 , M_3 , and M_4). Remaining members decrypt the received MLN message and obtain the new SK_1^0 , key-ID, and key counter.
- (5) SC_1 continues multicast payload data transmission, encrypted with the new subgroup key (seed) SK_1^0 .

3.3 Member Move

During a multicast session, a member may move from one subgroup to another due to two reasons: a subgroup is overloaded, so the system decides to relocate some members to other subgroups (may start new subgroups); in a wireless mobile environment, the mobility allows members to move from one subgroup to another subgroup without leaving the multicast session. For example, M_2 is moving from subgroup 1 to subgroup 3, as shown in Figure 2:

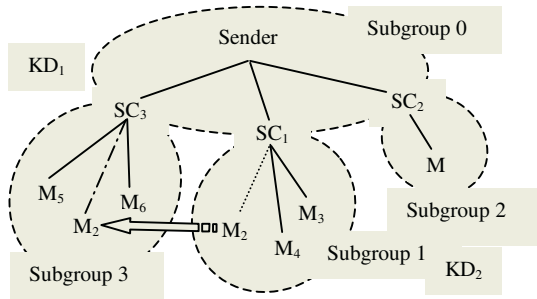


Fig. 2. Member move

- (1) SC_1 forwards M_2 's information (i.e. member identifier, member's public key, membership start and end time) to SC_3 . SC_1 does not re-key its subgroup key, but deletes M_2 from its CML and adds this member to its Moving Out Members List (MOML). Multicast payload data transmission is not halted in source subgroup since no re-keying operations are performed.
- (2) If M_2 is in SC_3 's MOML, it means that M_2 is a former member of subgroup 3 and her key-ID is still valid. SC_3 just deletes M_2 from its MOML, adds her to its CML, and multicasts an MJN message to its subgroup. M_2 keeps the subgroup key (e.g. SK_3^p), the key-ID and the key counter (e.g. p) since she left subgroup 3 (see Table 3). Once M_2 receives SC_3 's MJN message, she can calculate the current subgroup key SK_3^n by comparing the received key counter and her stored one, i.e. $SK_3^n = F^{n-p}(SK_3^p)$ ($n \geq p$). Existing members, M_5 and M_6 , drop the MJN message after verifying that the received key-ID and key counter are same with their local ones. Therefore, no subgroup re-keying operations are performed and the multicast payload data transmission is not halted in the destination subgroup.

The Group Key (GK)	Member's public key
The shared unicast key with SC_i	
Current subgroup key for subgroup i : SK_i^n ; key-ID of SK_i^0 ; key counter: n .	
The latest authenticated TESLA MAC key (The key chain is created by SC_i).	
Keying material of visited subgroups	Subgroup key in subgroup j when left: SK_j^p ; key-ID of SK_j^0 ; key counter (p).
	...

Table 3. Storage for a member in subgroup i

- (3) If there is no entry of M_2 in SC_3 's MOML, SC_3 negotiates a unicast key with M_2 and adds M_2 to its CML. Then, SC_3 compares the start time of the current subgroup key SK_3^n (i.e. TK_3) with M_2 's membership start time.

Case 1: M_2 's membership start time is earlier than TK_3 . Re-keying SK_3^n is not required and SC_3 only needs to send SK_3^n , key-ID, and key counter to M_2 via secure unicast.

Case 2: M_2 's membership start time is later than TK_3 . To keep membership backward secrecy, SC_3 has to perform the member join operation, as presented in Section 3.1, to accept M_2 into its subgroup.

Because the member move operation does not require the source subgroup to perform re-keying immediately, a member may accumulate subgroup keys as she visits different subgroups. Thus, one member leave operation may cause more than one subgroup to perform re-key operations to keep membership forward secrecy. That is why a SC needs to multicast a message to subgroup 0 to notify

all other SCs that a member in its subgroup is leaving, as shown in step 2 of the member leave operation. Each SC then checks its CML and MOML: if the leaving member is in its CML or MOML, the SC needs to re-key its subgroup key via the member leave operation to achieve membership forward secrecy; otherwise, no re-keying operations are required. Every time a SC performs a member leave operation, it also resets its MOML since a new key seed has been generated and members in MOML are no longer have the valid key seed.

To limit the maximum amount of time that SK_i can be held by a member outside subgroup i , each SC_i maintains a timer for current subgroup key SK_i^n . At $t = T_i$ (a threshold value), SK_i^n is updated via a member leave operation and the timer is set to zero. This is one type of periodic re-keying [19]. At this point, no group member outside of subgroup i has a valid SK_i^n .

3.4 Key Redistribution

In the proposed scheme, if member M_i cannot decrypt received multicast data correctly, it means M_i does not obtain the latest keying material (i.e. M_i missed latest MJN or MLN messages). The key redistribution is handled as follows:

- (1) M_i first waits a threshold of time for any MJN or MLN messages from the SC. The threshold value can be set according to different applications. For a multicast group whose membership dynamics is high, M_i can expect a short wait for the next MJN or MLN message. If the received message is MLN, the member obtains the latest subgroup key by decrypting this message using the unicast key. If the message is MJN, the member acquires the latest keying material (i.e. the current key-ID and key counter).
- (2) If the threshold of waiting time is reached, the member sends the SC a request (encrypted and authenticated with the shared unicast key) for an MJN message. Here, the benefit is that an MJN message is sent to the subgroup upon the first request. Other members in this subgroup, who also have not received the latest keying materials, can obtain this notification message without sending duplicate requests.
- (3) If the received key-ID is the same as M_i 's stored key-ID, it means that M_i only missed several MJN messages since her key-ID is still valid. Thus, M_i only needs to perform F on its stored subgroup key several times to acquire the current subgroup key. For example, the key counter from the SC is u and M_i 's key counter is v ($v < u$), M_i performs $F^{u-v}(SK_j^v)$ to get the current subgroup key SK_j^u .
- (4) If two key-IDs are not the same, it means that the member's key-ID is not valid anymore. The member must have missed at least one MLN message. In this case, the member needs to obtain the current subgroup key from the SC via secure unicast.

3.5 Data Transmission and Authentication

We propose two methods for dual encryption of multicast payload data. Both of them can hide data from third-parties (i.e. SCs and KDs). Method 1:

- (1) The sender encrypts the multicast packet P with GK , then encrypts the result with the current subgroup key in subgroup 0 (e.g. SK_0^m) to get the ciphertext $C = E_{SK_0^m}(E_{GK}(P))$, and finally multicasts C to subgroup 0.
- (2) SC_i decrypts C with SK_0^m , and then re-encrypts the result with its current subgroup key SK_i^n to get ciphertext $C' = E_{SK_i^n}(E_{GK}(P))$. SC_i multicasts C' to its subgroup.
- (3) Members in subgroup i first decrypt the received ciphertext C' with their current subgroup key SK_i^n ; then, members use GK to decrypt the result and obtain the original packet $P = D_{GK}(D_{SK_i^n}(C'))$.

Method 2: The sender encrypts each multicast packet with a unique Data Encryption Key (DEK). SCs just forward encrypted packets to their subgroups without decrypting and re-encrypting multicast payload data. Instead, the dual encryption scheme presented in method 1 is used to distribute each DEK to group members. This data transmission method saves encryption and decryption bandwidth on both SCs and members since DEK s (each 128-bit long) are smaller than multicast payload data packets (several hundred bytes each).

As for data authentication, TESLA's advantages of loss tolerance and high efficiency make it an ideal solution to be adopted in the proposed scheme to handle unreliable multicast service and high dynamic membership updates. A detailed description of implementing TESLA in secure multicast protocols can be found in [17].

4 Security and Benefits Analysis

4.1 Security

Group Access Control. Group access control is fulfilled through digital credentials. Only those members who purchase credentials can enter the multicast group and obtain keying material from key distributors and subgroup controllers. KDs and SCs can verify the sender's digital signature on those credentials using the sender's public key. Moreover, digital credentials are one-time-use, which prevent duplicate join requests from malicious members.

Group Membership Control. The proposed group key management scheme archives membership backward secrecy for the member join operation. The new member cannot decrypt multicast payload data sent before she joins the group since she cannot obtain former subgroup keys (i.e. it is impossible for the new member to get SK_i^0, \dots, SK_i^n from SK_i^{n+1} because one-way has function F provides perfect backward secrecy).

As Mittra points out, the method used to process member join (i.e. multicasting a new subgroup key to existing subgroup members encrypted under the old subgroup key) has the weakness that it sets up a chain of subgroup key whereby a compromise in one link of the chain results in a compromise of all the following subgroup keys [14]. The proposed member join scheme also creates a

subgroup key chain due to the continuous applying the one-way hash function F on the current subgroup key seed. To break this subgroup key chain, each SC can re-key the subgroup key periodically, which is fulfilled through the member leave scheme.

Membership forward secrecy for the member leave operation is also fulfilled. If a member leaves the subgroup, a new subgroup key seed SK_i^0 is generated to replace the current subgroup key and is shared by remaining members. Therefore, the former member who left the group cannot decrypt future encrypted messages because she does not have the new subgroup key seed. As discussed in Section 3.3, membership forward secrecy for the member move operation is achieved when moving members finally leave the multicast group.

Secure Key Distribution and Redistribution. For the member join operation, the new subgroup key is sent by an SC to the new member via secure unicast; outsiders, therefore, cannot obtain the new key. Existing members obtain the latest subgroup key by performing the one-way hash function locally without any secret key transmission. The MJN message is in plaintext format since it does not provide sufficient keying material for outsiders. As for the member leave operation, the new subgroup key is encrypted under unique unicast keys and multicasted to the remaining members, thus it is unobtainable to outsiders. The member move and key redistribution operations are also secure because they are based on the secure member join and leave operations.

Secure Data Transfer. Both of two data transmission methods are secure. For method 1, multicast data are encrypted dually under GK and a subgroup key. Only current group members can decrypt the multicast payload successfully because non-members (including SCs, KDs, and former members) do not have GK or current subgroup keys.

For method 2, multicast data are encrypted by a per-packet DEK . Because every DEK is securely distributed to current group members via the dual encryption method, SCs and KDs cannot obtain any DEK . Former group members cannot acquire any DEK s after they leave the multicast group.

Although the whole group shares a group-wide GK , the member who resides in one subgroup cannot decrypt other subgroups' messages because a different subgroup key is used for each subgroup.

4.2 Efficiency

The benefit of multicasting a Member Join Notification message in plaintext format is the elimination of the secret key transmission. Thus, it remove the vulnerability of the traffic containing the new subgroup key to cryptanalysis. Considering that most one-way hash functions work faster than block ciphers (e.g. MD5 and SHA-1 are 2-8 times faster than AES and DES [2, 18]), the member join operation of the proposed group key management scheme is faster than that of existing protocols.

More important, the MJN message contains no confidential contents so that it does not need to be encrypted (though it still needs to be authenticated via the TESLA protocol). In most existing protocols, the more member join operations, the more encryption/decryption operations and secret key transmissions involved. However, in the proposed scheme, member join operations provide opportunities for those members who do not have latest subgroup key to obtain the latest keying material (i.e. plaintext MJN messages). Thus, efficient key redistribution is fulfilled based on the efficient member join method.

4.3 Scalability

A scheme is scalable if the overhead involved in key updates, data transmission, and encryption does not depend on the size of the whole multicast group [14]. The proposed group key management scheme achieves scalability through subgrouping of multicast members. The subgrouping localizes the effect of group membership changes (e.g. member join) to one subgroup.

In member leave and move operations, more than one subgroup could be involved in re-keying operations. As mentioned in Section 3.3, the source subgroup does not re-key its subgroup key immediately but adds the moving member in its MOML. The MOML is reset after a threshold of time (periodic re-keying) or by member leave operations. When the former member returns to the subgroup and she is still in the MOML, the SC does not need to re-keying its subgroup. Thus, maintaining an MOML decreases the number of re-keying operations. However, if a leaving member exists in many SCs' MOMLs, this member's leave request will then cause more than one subgroup to re-key subgroup keys. Therefore, the threshold time may be set according to the group membership dynamics.

5 Secure Multicast for Multimedia Data

Secure multimedia multicast covers many commercial applications, such as pay-per-view Internet TV, Radio, and Video. Multimedia data have different characteristics from text data (i.e. information rate is very high while information value is low); therefore, low security levels but fast encryption algorithms are preferred for secure multimedia multicast. The *copyright protection* problem is another challenging issue: the downloaded digital contents can be easily copied and distributed to unauthorized parties by some multicast group members (so-called *traitors* [1]). In this section, we present how to deploy fast MPEG (Moving Picture Experts Group) encryption algorithms and digital rights management (DRM) systems to the proposed group key management scheme.

5.1 Fast MPEG Encryption Algorithms

The default encryption algorithms used in the proposed scheme to encrypt multicast payload data are block ciphers (e.g. AES [2]). However, if multicast payload

data are real-time video bit-streams, using conventional block ciphers adds a large amount of processing overhead because of the high data rate of video.

To reduce the encryption and decryption processing overhead, several MPEG video encryption algorithms have been proposed [20]. These algorithms can be divided into roughly two categories: algorithms for uncompressed bit-streams and algorithms for compressed bit-streams. Because the algorithms for uncompressed bit-streams are performed before coding, they decrease compression efficiency and also change original MPEG architecture; hence, algorithms for compressed bit-streams are preferable.

Zhang *et al* [20] have proposed a fast encryption algorithm based on the MPEG-2 and MPEG-4 bit-stream syntax. We adopt this algorithm into the proposed group key management scheme because it provides strong security to prevent from known-plaintext attack, while simultaneously maintaining low processing overhead, visual acceptance, constant bit rate, and bit-stream compliance. It also provides flexible security solutions to meet different security requirements by changing the value of several parameters.

5.2 Digital Rights Management Systems

The need for strong digital rights management technology has increased due to vast improvements in streaming media, compression technology, and the Internet. The downloaded high-quality digital media can easily be copied and distributed. Consequently, content providers face serious problems in protecting their rights over this digital media [13].

We use Microsoft Windows Media Digital Rights Management 9 Series (WM-DRM 9) as an example to demonstrate how to deploy a DRM system in the proposed group key management scheme. Figure 3 depicts the work diagram of WMDRM 9 and the detailed procedure can be found at [13]. WMDRM 9 involves five entities: content owner, content packager, content distributor, consumer, and license issuer. To deploy WMDRM 9 in the proposed group key management scheme, the sender of the multicast system becomes the content owner and content packager, which encodes digital files into Microsoft Media files and packages them (including encrypt files under a license key and we still call it *GK*). SCs operate as content distributors and are still responsible for managing subgroups. KDs serve the multicast group as license issuers. Multicast group members are, of course, consumers.

The original member join procedure also need a change. There is no contact between the new member and KDs when processing the member join request, i.e. *GK* is not sent to the new member by any KDs at that point in time. After SC continues multicast data transmission, existing members play the received file as expected because a valid license is already on their computers. For the new member, a valid license is not found, so a license request is made to the license acquisition URL, which directs to one of the KDs (license issuers). The KD sends the license (containing *GK*) to the new member. Finally, the new member plays the received file.

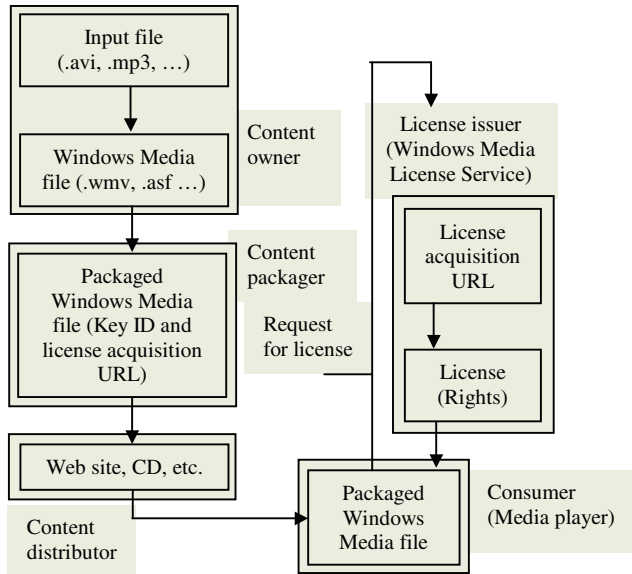


Fig. 3. Windows Media Digital Rights Management work diagram

Note that consumers can share Windows Media files but each recipient must have a separate license to play them, i.e. licenses are bound to the machine that received them and cannot be shared [13]. Therefore, by deploying digital rights management systems, the proposed group key management scheme is able to prevent unauthorized parties from playing the received digital contents, which are illegally copied from multicast group traitors.

6 Conclusions

The popularity of multicast communications has grown considerably with the wide use of the Internet; therefore, the need to secure multicast communications is critical. This paper focuses on the designs of an efficient and scalable group key management scheme and its applications in secure multimedia multicast. Considering that members of the one-to-many multicast are normally low-end machines with limited resources, this paper provides efficient security solutions, such as efficient member join and move methods, efficient key redistribution, efficient authentication, etc. Thus, the proposed group key management scheme is an ideal solution for resource-constraint equipments (e.g. set-top-boxes and wireless hand-held PDAs). Combined with fast MPEG encryption algorithms and digital rights management systems, the proposed scheme provides a complete and efficient solution for delay-sensitive secure multimedia multicast (e.g. live audio/video multicast).

References

- [1] B. Chor, A. Fiat, M. Naor, and B. Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):257–270, May 2000.
- [2] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.
- [3] B. DeCleene, L. Dondeti, S. Griffin and T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang. Secure group communications for wireless networks. In *Proceedings of IEEE MILCOM 2001*, Mclean, VA, October 2001.
- [4] S. Deering. Host extensions for ip multicasting. IETF RFC 1112, August 1989.
- [5] L. Dondeti, S. Mukherjee, and A. Samal. Survey and comparison of secure group communication protocols. Technical report, University of Nebraska-Lincoln, 1999.
- [6] L. Dondeti, S. Mukherjee, and A. Samal. Scalable secure one-to-many group communication using dual encryption. *Computer Communications Journal*, pages 1681–1701, November 2000.
- [7] A. M. Eskicioglu. Multimedia security in group communications: Recent progress in key management, authentication, and watermarking. *Multimedia Systems*, 9:239–248, September 2003.
- [8] T. Hardjono, B. Cain, and N. Doraswamy. A framework for group key management for multicast security. IETF Internet Draft, draft-ietf-ipsec-gkmframework-03.txt, August 2000.
- [9] T. Hardjono, B. Cain, and I. Monga. Intra-domain group key management protocol. IETF Internet Draft, draft-ietf-ipsec-intragkm-03.txt, September 2000.
- [10] S. Kent and R. Atkinson. Security architecture for the internet protocol. Network Working Group, RFC 2401, November 1998.
- [11] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication. Network Working Group, RFC 2104, February 1997.
- [12] P. McDaniel, H. Harney, P. Dinsmore, and A. Prakash. Multicast security policy. IETF Internet Draft draft-irtf-smug-mcast-policy-01.txt, November 2000.
- [13] Microsoft. Microsoft windows media - digital rights management. At <http://www.microsoft.com/windows/media/drm.aspx>.
- [14] S. Mitra. Iolus: A framework for scalable secure multicasting. In *Proceedings of ACM SIGCOMM'97*, pages 277–288, Cannes, France, September 1997.
- [15] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Tesla: Multicast source authentication transform introduction. IETF Internet Draft, draft-msec-tesla-intro-01.txt, October 2002.
- [16] A. Perrig, R. Canetti, and B. Whillock. Tesla: Multicast source authentication transform specification. IETF Internet Draft, draft-ietf-msec-tesla-spec-00.txt, October 2002.
- [17] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, pages 521–534, 2002.
- [18] B. Preneel, V. Rijmen, and A. Bosselaers. Recent developments in the design of conventional cryptographic algorithms. In *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science, pages 106–131. Springer-Verlag, 1998.
- [19] S. Setia, S. Koussih, and S. Jajodia. Kronos: A scalable group re-keying approach for secure multicast. In *Proceedings of 2000 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000.
- [20] C. N. Zhang and Y. Wu. A fast encryption algorithm for mpeg videos. In *the Third International Conference on Intelligent Multimedia Computing and Networking*, pages 26–30, Cary, North Carolina, USA, September 2003.

Author Index

- Ali, Y. 43
Argyroudis, P.G. 246

Baier, H. 112
Baykal, N. 306
Bicakci, K. 306

Cánovas, Ó. 297
Cock, D. De 1

Domingo-Ferrer, J. 180

Efsthathiou, E.C. 260
Esparza, O. 28

Florio, L. 173
Forné, J. 28

Gaj, K. 232
Giorgini, P. 98
Gómez-Skarmeta, A.F. 297
Gritzalis, D. 287

Han, K. 350
Hühnlein, D. 314
Hwang, Y.H. 322

Ine, S.R. 350

Kambourakis, G. 287
Kang, S.-k. 350
Karatsiolis, V. 126
Kim, C.H. 322
Kim, K. 350
Koga, S. 85
Konstantinou, E. 335

Lambrinoudakis, C. 149
Lee, P.J. 71, 322
Lee, S. 350

Li, Z. 364
Leros, A. 149
Libert, B. 57
Lioy, A. 14
Lippert, M. 126
Lopez, D.R. 173
López, G. 297

Malagon, C. 173
Marian, M. 14
Mark, B.L. 232
Massacci, F. 98
Mezzetti, N. 191
Mitchell, C.J. 205
Moltchanova, N. 14
Montenegro, J.A. 160
Moya, F. 160
Muñoz, J.L. 28
Mylopoulos, J. 98

O'Mahony, D. 246

Pala, M., 14
Pashalidis, A. 205
Platis, A. 149
Polyzos G.C. 260
Preneel, B. 1

Quisquater, J.-J. 57

Rossnagel, H. 274
Rouskas, A. 287
Russell, S. 135
Ryou, J.-C. 85

Sakurai, K. 85
Smith, S. 43, 218
Soriano, M. 28
Stamatiou, Y.C. 335
Straub, T. 112

Thomas, R.K. 232

Vanrenen, G. 218

Wiesmaier, A. 126

Wouters, K. 1

Yum, D.H. 71

Zannone, N. 98

Zaroliagis, C. 335

Zhang, C.N. 364

Zouridaki, C. 232